TECHNICAL REPORT RDMR-AD-16-01

JOINT COMMON ARCHITECTURE DEMONSTRATION (JCA DEMO) FINAL REPORT

Scott A. Wigginton
Aviation Development Directorate
Aviation and Missile Research, Development, and Engineering Center

July 2016

REPORT DOCUMENTATION PAGE	**Form Approved OMB No. 074-0188**

1. AGENCY USE ONLY	**2. REPORT DATE** July 2016	**3. REPORT TYPE AND DATES COVERED** Final

4. TITLE AND SUBTITLE Joint Common Architecture Demonstration (JCA Demo) Final Report	**5. FUNDING NUMBERS**
6. AUTHOR(S) Scott A. Wigginton	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Commander, U.S. Army Research, Development, and Engineering Command ATTN: RDMR-ADA-IN Redstone Arsenal, AL 35898-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER** TR-RDMR-AD-16-01
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT *(Maximum 200 Words)*

The Joint Common Architecture Demonstration (JCA Demo) project was the first in a series of planned experiments under the Joint Multi-Role (JMR) Technology Demonstrator (TD) Mission Systems Architecture Demonstration (MSAD) Science and Technology (S&T) effort. JCA Demo focused on maturating key standards, processes, and tools necessary for the acquisition of affordable, common, reusable avionics capabilities. These include the Future Airborne Capability Environment (FACE™) standard, JCA Functional Reference Architecture, Model-Based Engineering (MBE) and Architecture Centric Virtual Integration Process (ACVIP). Through a competitive model-based acquisition effort, two vendors were selected to develop a reusable software component that the Army successfully integrated and demonstrated on two undisclosed mission computers. The JCA Demo approach of learning by doing resulted in dozens of actionable lessons learned that matured each of the standards, processes, and tools necessary to tackle complexity and affordability.

14. SUBJECT TERMS Joint Common Architecture (JCA), Future Airborne Capability Environment (FACE™), Open Systems Architecture (OSA), Model-Based Engineering, Architecture Centric Virtual Integration Process, Reusable Software Component, Software Portability, Avionics	**15. NUMBER OF PAGES** 70
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** SAR

NSN 7540-**01**-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

ACKNOWLEDGMENTS

The Aviation Development Directorate (ADD) of the United States (U.S.) Army Aviation Missile, Research, Development and Engineering Directorate (AMRDEC) gratefully acknowledges the contributions of the following organizations in the performance of this effort:

- AMRDEC Aviation Engineering Directorate
- AMRDEC Software Engineering Directorate
- AMRDEC Aviation Systems Integration Facility
- Honeywell Aerospace
- Sikorsky Aircraft Corporation
- The Boeing Company

TABLE OF CONTENTS

TABLE OF CONTENTS (CONCLUDED)

Page

LIST OF ILLUSTRATIONS

LIST OF TABLES

I. INTRODUCTION

A. Problem Statement

The integration of mission equipment and software into Department of Defense (DoD) aircraft is increasingly inefficient and becoming unaffordable. This is largely due to the logarithmic growth in system complexity and rapid pace of change driven by technological advancement and the proliferation of sophisticated threats. Aircraft system capabilities based on vendor-defined architectures introduce peculiar interfaces with highly specialized dependencies, making component upgrade or replacement very expensive. Initial implementations may be economical and effective for a specific aircraft, but over time, they restrict capability improvements and hinder or prevent cross-platform commonality and fleet-wide efficiencies.

Minimizing the cost and elapsed time between the emergence of useful technologies and fielding operational capabilities are components of national comparative advantage. Plainly stated, the ability to incorporate technical advances faster than adversaries is part of the modern battlefield.

The role software contributes to capabilities in modern aircraft systems cannot be overstated. DoD systems increasingly rely on software to achieve their performance characteristics [1]. Software has become a dominant factor in system acquisition with nearly every aircraft function dependent on software [2], as shown in Figure 1. Software requirements growth for commercial aircraft will soon exceed affordability limits [3], as shown in Figure 2.

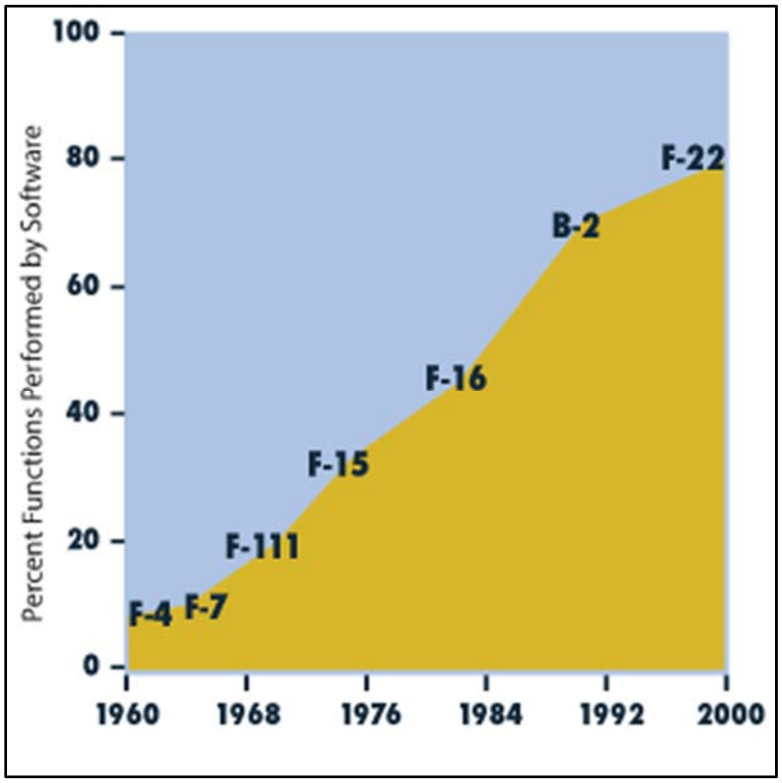


Figure 1. Combat Aircraft Functionality Requiring Software

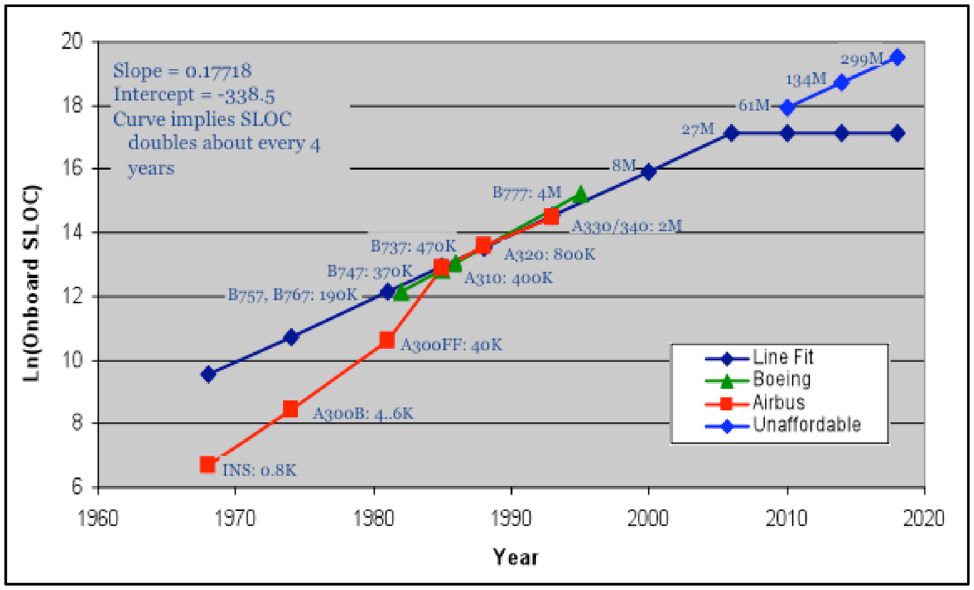


Figure 2. Estimated Onboard SLOC Growth

The Government Accountability Office (GAO) has numerous reports on complex aircraft systems that encountered difficulties with software, such as the AH-64D Apache Longbow [4], RAH-66 Comanche [1, 5], C-17 [6], F/A-18 [7], F-22 [1, 8], and F-35 Joint Strike Fighter [9-12] to name a few. Challenges with software have led to significant cost increases, schedule delays, fielded capabilities shortfalls, and catastrophic loss of life and property [13].

The National Aeronautics and Space Administration (NASA) recommends [14] that tackling the architecture of systems is the best way to effectively manage complexity. Fielding a next-generation rotary wing aircraft using existing architectural approaches is not affordable [15]. Proper application of emerging Open Systems Architecture (OSA) approaches and Model-Based Engineering (MBE) techniques are required to manage the complexity and achieve the desired level of commonality across the aviation enterprise.

B. Joint Common Architecture Concept

The Joint Common Architecture (JCA) is intended to define Reusable Software Components (RSCs) that reside on the mission computers of the vertical lift fleet. JCA Version 1.0 will comprise a functional description of the RSCs and is being sponsored by the Joint Multi-Role (JMR) Mission System Architecture Demonstration (MSAD) effort. The primary objective of JCA is to enable the procurement of affordable warfighting capability through the planned and strategic reuse of hardware and software assets across a Future Vertical Lift (FVL) Family of Systems (FOS) using a combination of data rights, platform abstraction, semantic precision, and functional allocation. JCA is an implementation and technology-independent conceptual framework, providing a common vision and taxonomy that is intended to be used as the starting point for the design and development of FVL and legacy upgrade avionics architectures. JCA follows OSA principles and leverages existing and emerging OSA related efforts to include the Future Airborne Capability Environment (FACE^{TM}) Technical Standard and uses a model-driven process and design principles.

C. Joint Common Architecture Demonstration Project

The JCA Demonstration (JCA Demo) project was established to verify the JCA concept and reduce risk for subsequent JMR MSAD efforts. JCA Demo investigated the enabling technologies and key characteristics of openness—modularity, portability and interchangeability.

JCA Demo resulted in the procurement of software components built to the same specification but acquired from multiple vendors. These components were integrated into multiple undisclosed Operating Environments (OEs) by a government integration team. An OE is the combination of computing hardware and supporting software that operate below the software application layer, the most common example of an OE is a mission computer. Additionally, the software was executed against a common scenario in a government laboratory. The demonstration sought to validate the following concepts:

- Functionality and data specified using JCA is sufficient to enable different implementations to provide the same desired capability interchangeably.
- Components adhering to the FACE Technical Standard possess sufficient portability as to allow their operation on multiple, disparate OEs.
- External interfaces adhering to the FACE data architecture promote openness, supports interchangeability, and result in interoperability.

II. BACKGROUND

A. Future Vertical Lift

FVL is a DoD initiative to define, develop, and field a fleet of next generation air vehicles that will ensure the United States' dominance in the vertical lift domain throughout the 21st century and beyond. FVL development and fielding offers the potential to significantly improve aviation combat capabilities and provide critical support to the joint warfighting community. FVL is envisioned as a family of aircraft defined by multiple payload classes with the potential for service-unique or specific variants. A major objective is to achieve meaningful commonality between the air vehicle classes, mission equipment packages, and support structure, while closing the capability gaps identified in the FVL Capability Based Assessment that evaluated future DoD rotary wing aviation requirements.

B. Joint Multi-Role Technology Demonstrator

The Army established the JMR Technology Demonstrator (TD) in 2010 to assess critical technologies for the FVL initiative. JMR TD is a Science and Technology (S&T) effort intended to demonstrate transformational vertical lift capabilities and prepare the DoD for decisions regarding the replacement of the current vertical lift fleet. It is composed of two main efforts. The first effort is the air vehicle demonstration, which includes the design, build, and test of enabling technologies and features for a next generation rotorcraft. The second effort is MSAD, which is investigating, maturing, and demonstrating the processes, tools, and standards necessary for the analysis, development, and qualification of an effective and affordable mission systems architecture for FVL. The knowledge gained from this effort will be used to inform Army development of requirements for the anticipated FVL program. The JMR MSAD effort has three areas of emphasis: OSA, MBE, and Architecture Centric Virtual Integration Process (ACVIP). Figure 3 shows MSAD efforts that include JCA Development and Demonstration, Objective Mission Equipment Package (MEP) Definition, Architecture Implementation Process Demonstrations (AIPD), and Mission Systems Architecture Capstone Demonstration (Capstone Demo).

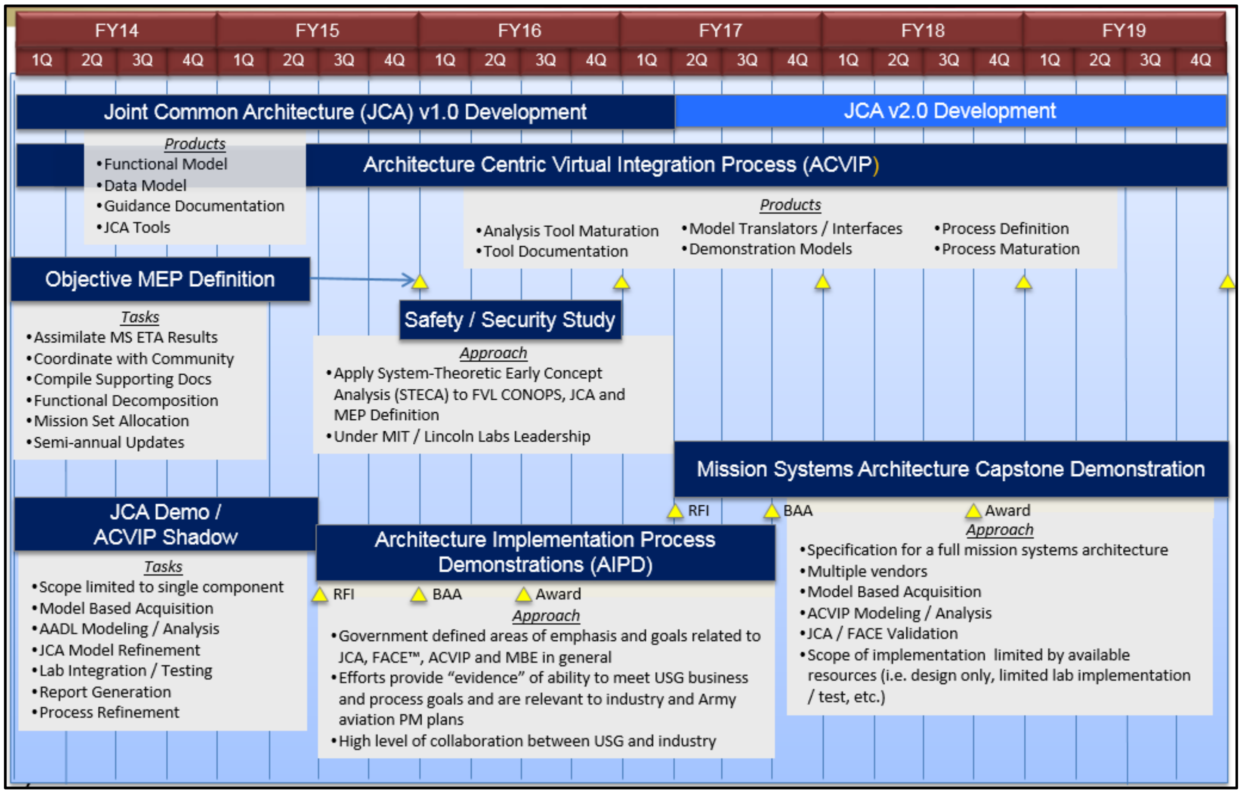

FY14
FY15
FY16
FY17
FY18
FY19
1Q 2Q 3Q 4Q
Joint Common Architecture (JCA) v1.0 Development
JCA v2.0 Development
Products
•Functional Model
•Data Model
•Guidance Documentation
•JCA Tools
Architecture Centric Virtual Integration Process (ACVIP)
Products
•Analysis Tool Maturation
•Tool Documentation
•Model Translators / Interfaces
•Demonstration Models
•Process Definition
•Process Maturation
Objective MEP Definition
Tasks
•Assimilate MS ETA Results
•Coordinate with Community
•Compile Supporting Docs
•Functional Decomposition
•Mission Set Allocation
•Semi-annual Updates
Safety / Security Study
Approach
•Apply System-Theoretic Early Concept Analysis (STECA) to FVL CONOPS, JCA and MEP Definition
•Under MIT / Lincoln Labs Leadership
Mission Systems Architecture Capstone Demonstration
RFI
BAA
Award
Approach
•Specification for a full mission systems architecture
•Multiple vendors
•Model Based Acquisition
•ACVIP Modeling / Analysis
•JCA / FACE Validation
•Scope of implementation limited by available resources (i.e. design only, limited lab implementation / test, etc.)
JCA Demo / ACVIP Shadow
Tasks
•Scope limited to single component
•Model Based Acquisition
•AADL Modeling / Analysis
•JCA Model Refinement
•Lab Integration / Testing
•Report Generation
•Process Refinement
Architecture Implementation Process Demonstrations (AIPD)
RFI
BAA
Award
Approach
•Government defined areas of emphasis and goals related to JCA, FACE™, ACVIP and MBE in general
•Efforts provide "evidence" of ability to meet USG business and process goals and are relevant to industry and Army aviation PM plans
•High level of collaboration between USG and industry

Figure 3. MSAD Efforts

C. Open Systems Architecture

OSA is the DoD preferred approach for implementing open systems [16], formerly known as the Modular Open Systems Approach (MOSA). OSA is a business and technical strategy to yield modular, interoperable systems that enables competition and innovation from different vendors. A fundamental premise associated with open systems is that one or more qualified third parties can add, modify, replace, remove, or provide support for a component of a system based on open standards and published interfaces for the component of that system. Successful OSA acquisitions result in reduced total ownership cost and enable systems to respond to changing user requirements. Reference 16 declares that the "essence of OSA is organized decomposition, using carefully defined execution boundaries, layered onto a framework of software and hardware shared services and a vibrant business model that facilitates competition."

OSA is composed of five fundamental principles:

1) Modular designs based on standards with loose coupling and high cohesion that allow for independent acquisition of system components

2) Enterprise investment strategies based on collaboration and trust that maximize reuse of proven hardware system designs and ensure that organizations spend the least to get the best

3) Transformation of the life cycle sustainment strategies for software intensive systems through proven technology insertion and software product upgrade techniques

4) Dramatically lower development risk through transparency of system designs, continuous design disclosure, and government, academia, and industry peer reviews

5) Strategic use of data rights to ensure a level competitive playing field and access to alternative solutions and sources across the life cycle

D. Integrated Modular Avionics

Integrated Modular Avionics (IMA) is an approach that divides avionics functionality into modules that can be developed and qualified incrementally for use in various aircraft. As opposed to a traditional federated avionics approach where avionics functions are tightly coupled on dedicated hardware components (typically, Line Replaceable Units (LRUs)), an IMA implementation includes shared resources that have been designed and verified to a set of safety and performance requirements. IMA offers the potential for significant reductions in the size, weight, and power of future mission systems as a result of the sharing and optimization of system resources. Such reductions translate to increased range, speed, and payload. DO-297 [17] provides guidance for assurance of IMA systems. IMA is a key enabler of RSCs, as defined by AC 20-148 [18]. Two key concepts to an IMA approach are incremental acceptance and compositional qualification.

Incremental acceptance refers to the idea that hardware and software components carefully selected and properly implemented can be accompanied by a partial qualification pedigree based on the achievement of a specific subset of all qualification objectives. When these achievements are documented, an airworthiness authority can endorse this partial pedigree for future use.

Compositional qualification refers to the strategy of the application of a partial qualification pedigree to the qualification of an integrated system that uses previously evaluated components. Incremental acceptance can apply at the level of software modules, software systems, or a hardware/software OE that supports the execution of software that provides avionics functionality that is developed separately.

E. Model-Based Engineering

MBE is described as "engineering practices in which models are the central and indispensable artifacts throughout a product's life cycle encompassing concept, development, deployment, operation, and maintenance" [19]. MBE is distinguished from traditional engineering practices primarily through its use of models to convey information. The goal of this approach is to improve communication, quality, and productivity that result from linked interactions conveying precise, detailed information. MBE is not a new concept nor is it a single solution to the engineering problems that exist today. The increasing level of integration and complexity of mission systems anticipated for the next generation of aviation platforms will require a level of specification, analysis, and awareness that is not achievable with existing document-based engineering practices. Model-based requirements, architecture specification, design, code generation, verification and validation activities, and advanced forms of analysis, including virtual integration, will be necessary. This will enable automated identification of issues and defects that historically have not been identified until later in the life cycle when these issues are significantly more difficult to resolve and much more costly to correct. MBE provides a mechanism for dealing with the complexity of software intensive systems.

ACVIP is a particular type of MBE focused on performing architecture centric analysis and uncovering issues prior to the implementation of hardware/software components or supporting trade-off analyses when considering system upgrades. This is a radically different approach to systems/software engineering that enables incremental virtual integration, verification, and certification processes to ensure system validity early and throughout the life cycle, minimizing defect insertion and propagation. ACVIP focuses on architecture analyses in the areas of requirements, safety, security, resources, and assurance and permits the virtual integration, verification, and generation of systems. It relies on a semantically precise language, such as the Architecture Analysis and Design Language (AADL), to model the architecture. The goal of ACVIP is to improve quality and reduce cost and schedule to develop software-intensive, safety- and security-critical systems. During the JCA Demo, parallel ACVIP Shadow effort was conducted by Carnegie Mellon University (CMU) Software Engineering Institute (SEI) and Adventium Labs, as shown in Figure 4. It focused on requirements, safety, and timing analysis of the component and its integration into the larger system. Details and results from those efforts are in References 20 through 24.

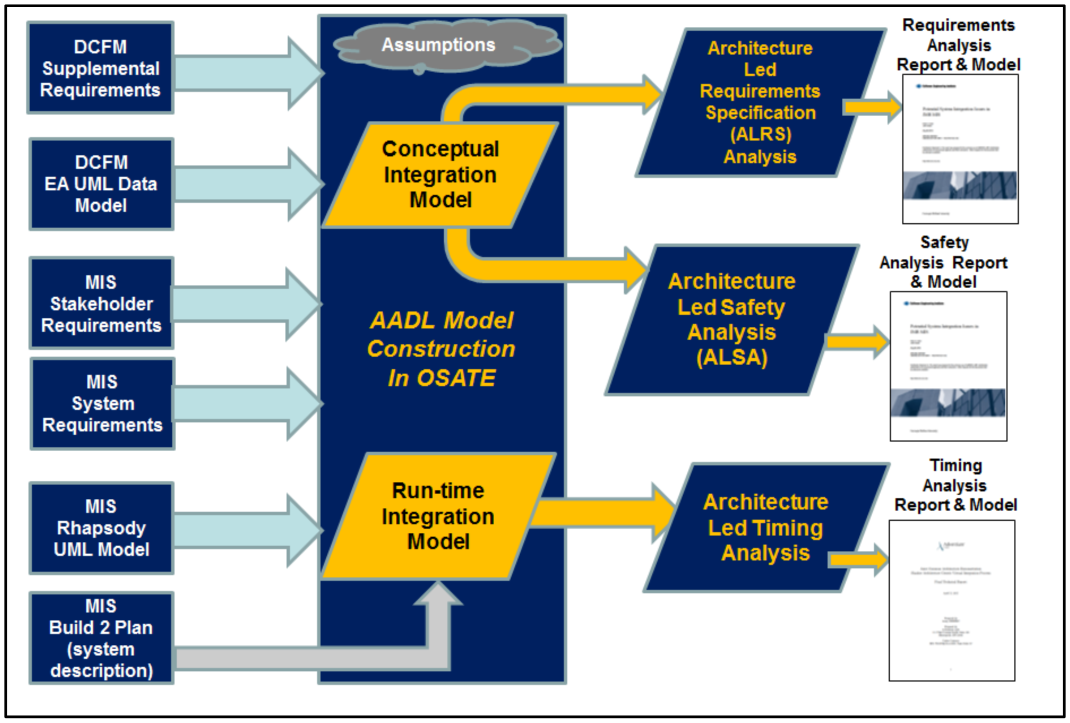

DCFM Supplemental Requirements
DCFM EA UML Data Model
MIS Stakeholder Requirements
MIS System Requirements
MIS Rhapsody UML Model
MIS Build 2 Plan (system description)
Assumptions
Conceptual Integration Model
AADL Model Construction In OSATE
Run-time Integration Model
Architecture Led Requirements Specification (ALRS) Analysis
Architecture Led Safety Analysis (ALSA)
Architecture Led Timing Analysis
Requirements Analysis Report & Model
Safety Analysis Report & Model
Timing Analysis Report & Model

Figure 4. ACVIP Shadow Effort

F. Future Airborne Capability Environment

The FACE Technical Standard [25] establishes and standardizes a common software environment that supports portability of software applications across systems. It is based on IMA and OSA principles and utilizes current widely adopted industry standards. In contrast to previous OSA initiatives which offer only general guidance on designing open systems, the FACE Technical Standard clearly describes the reference architecture and specifies the key interfaces that enable a product line approach to software development, as shown in Figure 5. Furthermore, the FACE approach is built around a business strategy with policies and procedures for establishing and maintaining a marketplace of FACE software components that can be reused by multiple systems.

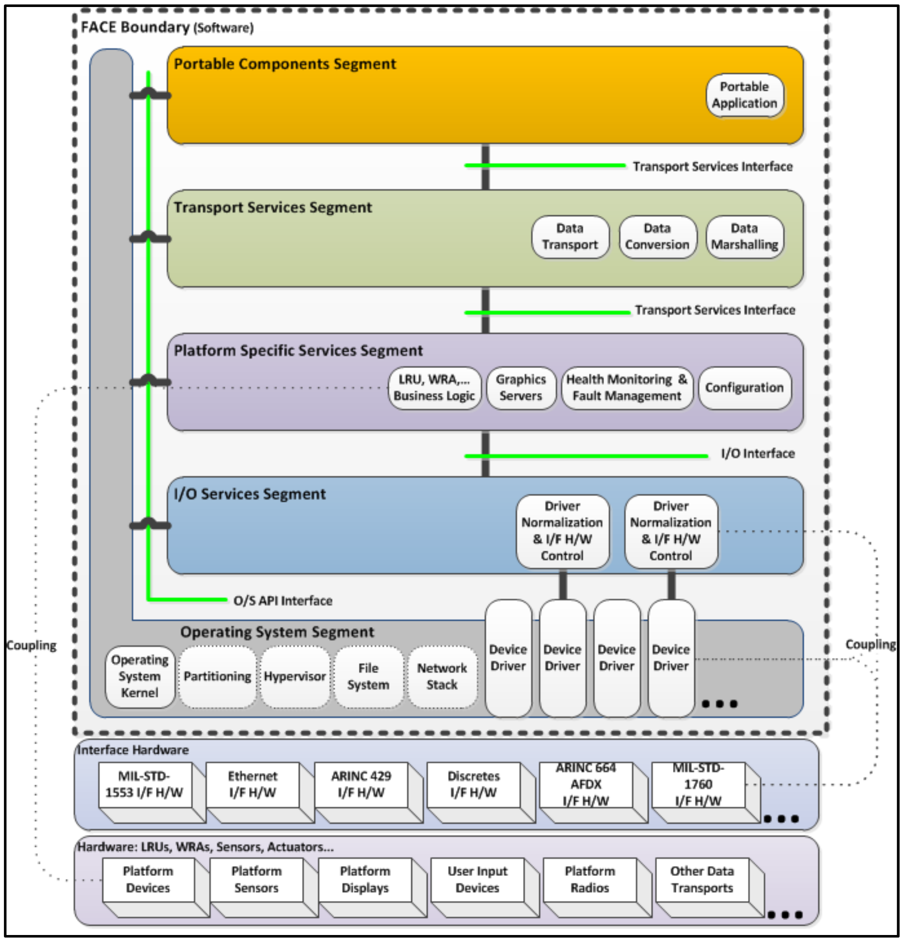


Figure 5. FACE Architectural Segments

G. Joint Common Architecture

The JCA is considered a Functional Reference Architecture (FRA). As such, JCA is a government-owned, implementation, and technology-independent conceptual framework that is intended to be used as the starting point for the design and development of FVL and legacy upgrade avionics architectures. The JCA provides a conceptual description of a set of generic avionics subsystems (typically, LRUs) and a functionally decomposed mission computing subsystem comprising a functional model and a semantic data model. The mission computer subsystem is the current focus of JCA development efforts and is expected to be required to host FACE software. JCA defines the Mission Level Capabilities (MLCs) and the constituent Low-Level Capabilities (LLCs) that will be composed into definitions of procurable RSCs hosted on the mission computer. The MLC definitions include functional descriptions and the identification of data provided and required. As a conceptual level description, JCA identifies the high-level functionality of software enabled capabilities inclusive of the data provided and required by that functionality. The functional description provides the boundary of desired modularity while the identified data provides the context for the functionality. The data contained within the JCA FRA provide the basis for defining a government-owned software product line of RSCs that support a fleet-wide business strategy of OSA implementation and strategic reuse.

The JCA development is an evolving and maturing process, as shown in Figure 6, consisting of a collaborative mix of government and industry performed activities. While the responsibility for development of the JCA FRA lies with the government, development of the actual software component is expected to be an industry function. The government will define the needed capabilities, the organizational and functional boundaries as well as the definitions and specification of the component, and its interfaces at a conceptual and logical level. The process and methods for accomplishing this function is guided by an overarching and systematic approach to commonality and reuse. The resulting JCA component definitions will be managed by the government and serve as the basis for a product line approach that provides necessary software enabled capabilities to vertical lift platforms in a managed and consistent manner. JCA is focused on the definition, specification, and interaction of the component at its boundaries. This definition is provided to industry to guide and inform software development activities.

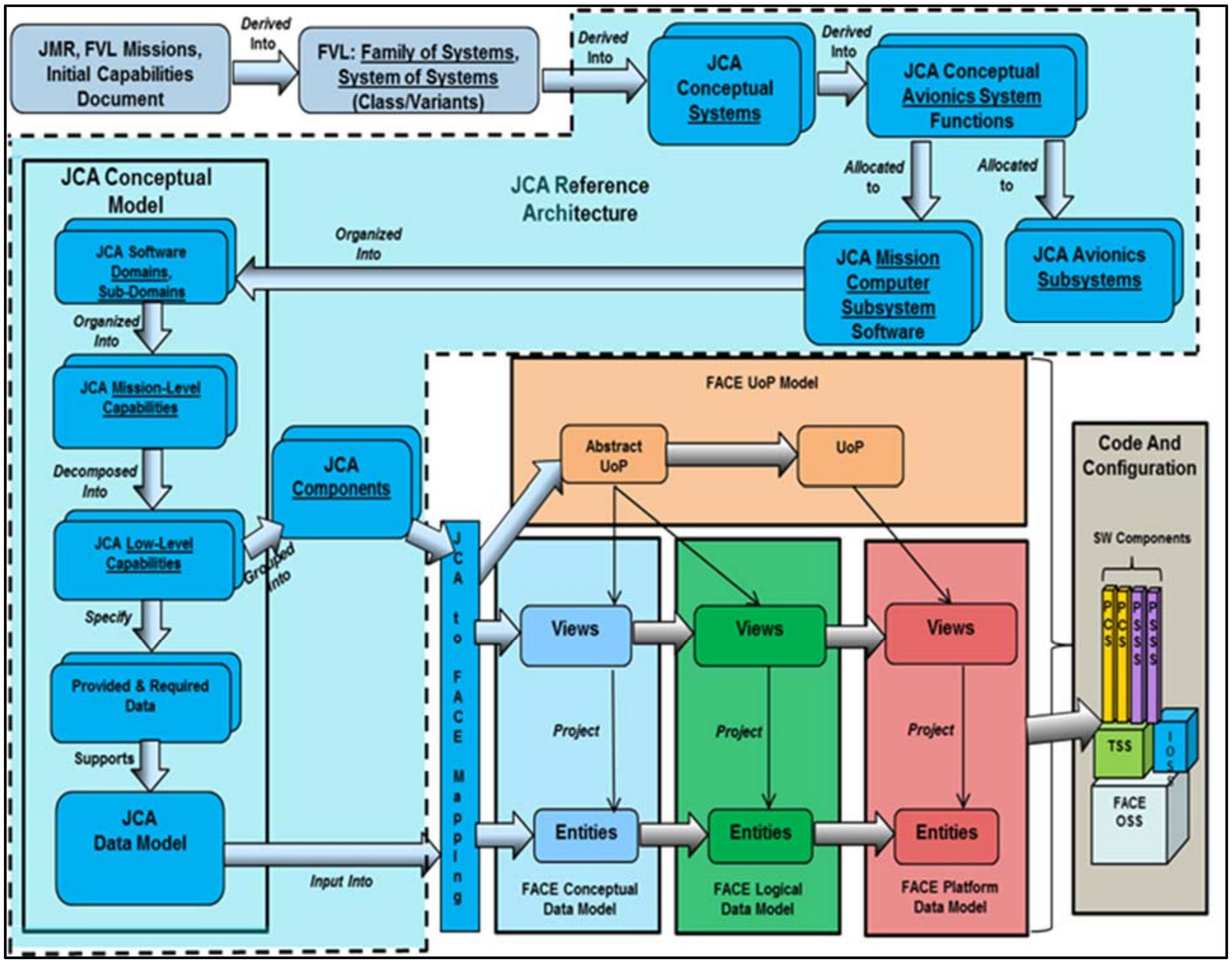


Figure 6. JCA Development Process

Avionics is the focus of JCA as it is anticipated to be a key areas within the FVL where opportunities for commonality and reuse exist. Army aviation currently reaps the benefits of reusing technically mature subsystems across its fleet of rotary wing aircraft, but the practice of reusing software capabilities is rare, and where it has occurred, it has been unplanned and inefficient. The reuse of software components in safety critical systems is a very immature practice and frequently thwarted by the tight coupling of software capability to platform specific implementations. Reuse for FVL at both the subsystem and component levels necessitates a common OE that enables efficient software portability and reuse. As a result, the United States (U.S.) Army Aviation and Missile Research, Development, and Engineering Center (AMRDEC) joined with the Navy as a founding member of the FACE Consortium. While the FACE Technical Standard is critical to JCA, it only addresses a portion of the overall scope of the problem.

III. METHODS AND RESULTS

The JCA Demo was conducted to validate the JCA concept, exercise the FACE standard and tools, and reduce risk for follow on MSAD efforts.

A. Experiment Design

The OSA Contract Guidebook [16] states that achievement of the five OSA principles requires that one or more qualified third parties be able to add, modify, replace, remove, or provide support for a component of a system, based on open standards and published interfaces for the component of that system. Therefore, the approach taken for JCA Demo was to procure a single software component from multiple vendors for integration into multiple OEs unknown to the vendors. This would demonstrate integration of multiple versions of the same component into multiple systems (a step beyond the OSA definition). The use of open standards and published interfaces was achieved by requiring alignment to the FACE standard with a model-based specification for a component generated from the JCA process. To ensure integrity of the process and maximize the lessons learned, the government served as Systems Integrator, and interactions with the vendors was tightly controlled. Once integrated into the OEs, the components were executed against a common mission scenario. The expected outcome would be to observe correct functionality against the component's requirements, regardless of who provided the component or what system it operated upon. The JCA Demo emphasized processes, tools, and lessons learned over the performance of the procured components.

Each vendor was required to deliver a Reusable Verification Component (RVC) as recommended in the Army handbook on airworthy reuse of FACE software [26] to ensure correct operation of the software component on any of the candidate OE's. The RVC builds on the concept of a portable software test harness to provide the full suite of capability required to verify that a software component satisfies all of its performance requirements regardless of computing environment. The RVC would contain test procedures and documentation for a Systems Integrator to execute on the target hardware.

The U.S. Army's FACE Verification Authority (VA) would evaluate the delivered software components for conformance to the FACE Technical Standard. The FACE Shared Data Model (SDM) Version 2.0 was not released in time for inclusion in the specification. As a result, it was known that the data model provided by the government would not pass that aspect of FACE conformance; however, all other aspects of FACE conformance were expected to pass.

The documentation, tracking, management, and resolution of lessons learned was a core element of JCA Demo. The mantra that processes, tools, and lessons learned are more important than component performance was applied throughout the execution of this study.

B. Specification Process

An S&T Situational Awareness (SA) system called Modular Integrated Survivability (MIS) was selected as the target system for JCA Demo with the MIS team performing the systems integration. The MIS system resides within the Aviation Systems Integration Facility (ASIF) at the AMRDEC Software Engineering Directorate (SED) complex on Redstone Arsenal, AL. MIS is comprised of multiple OEs, multiple sensors, a Multi-Function Display (MFD), and software built to the FACE Standard. Based on the types of data available within MIS, a Data Correlation and Fusion Manager (DCFM) was selected as the software component to be procured and integrated during JCA Demo. Due to cost and schedule considerations, two OEs were chosen for the demonstration, one based on the VxWorks® Operating System and another on the LynxOS® Operating System. Composition information of the MIS system was withheld from the vendors until after component software was delivered.

The government used an earlier version of the JCA process, as shown in Figure 7, to decompose the track correlation functionality into a DCFM software component specification. The DCFM was specified as a FACE Unit of Portability (UoP), where its interface and behavior was captured using a model-based specification in the form of a FACE data model plus two behavior component interaction diagrams. The government also provided a minimal set of functional requirements for demonstration purposes only, as shown in Table 1. Appendix A includes an excerpt from the DCFM specification.

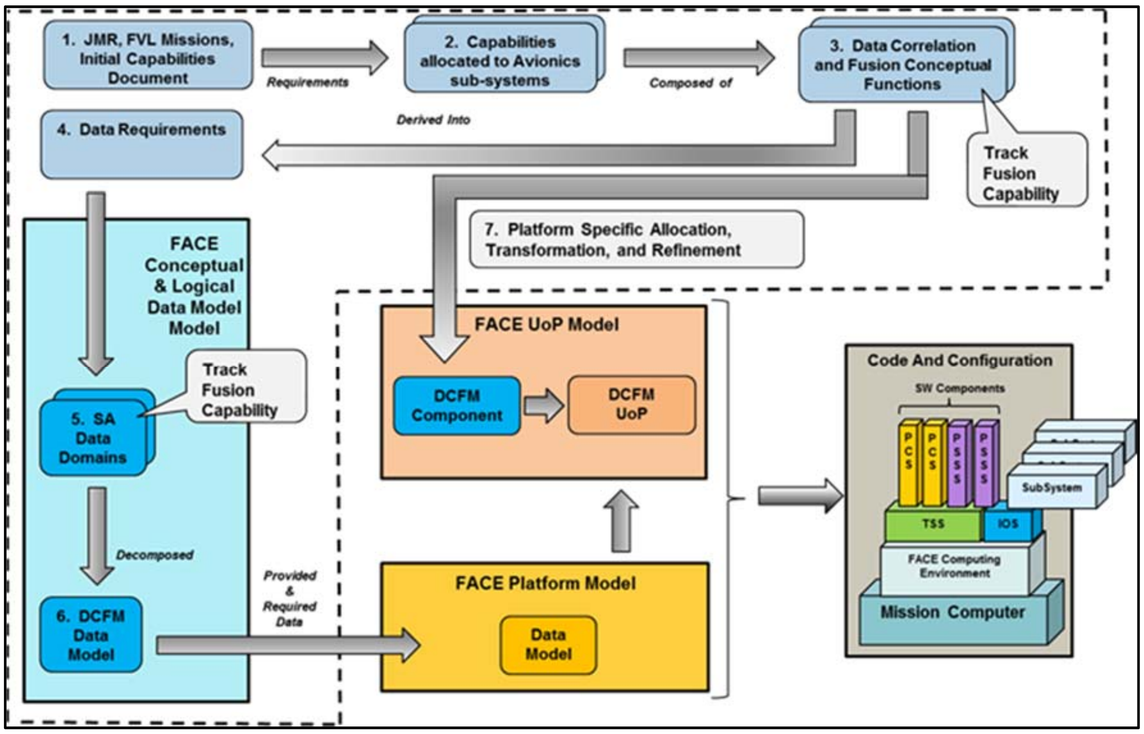


Figure 7. JCA Demo Specification Process

Table 1. JCA Demo Functional Requirements

ID	Requirement
JCAD_1	The DCFM shall analyze uncorrelated source tracks in order to identify a single correlated track believed to represent the same uncorrelated source tracks, combining the data from the duplicate uncorrelated source tracks as appropriate
JCAD_2	The DCFM shall analyze correlated source tracks in order to identify separate tracks, breaking the linkage of the correlated track with the uncorrelated source tracks, as appropriate
JCAD_3	The DCFM shall analyze tracks within 25 km of own-ship position
JCAD_4	The DCFM shall correlate or decorrelate 50 source tracks within 1 second
JCAD_5	The DCFM data model shall be used during the development of the software component
JCAD_6	The DCFM shall be built as a FACE UoP, as specified in the data model
JCAD_7	The DCFM shall have a verification statement provided by the candidate FACE Conformance Tool Suite for the FACE Edition 2.0

C. Procurement

A Broad Agency Announcement (BAA) was chosen as the solicitation method for JCA Demo. Prior to its release, a Request for Information (RFI) was published that included the draft BAA to determine the validity of the approach and the sufficiency of the model-based specification. Responses to the RFI were positive, resulting in minor modifications, and the BAA was released 6 months later. The OSA Handbook and draft FACE contract guide were used to generate Section L (Instructions to Offerors) and Section M (Evaluation Factors for Award) of the BAA. The BAA [27] is in Appendix B, and the following is a summary of the listed requirements:

- Developers were required to deliver a DCFM software component that met the requirements of the textual and model-based specification
- Developers were required to demonstrate how the DCFM would meet the requirements of the FACE Technical Standard
- Developers were required to utilize the candidate FACE Ecosystem tools, which included candidate versions of a Conformance Test Suite, Modeling Tools, and an Interface Definition Language (IDL) compiler
- Developers were required to deliver an RVC to provide unit test capability for any OE
- Developers were required to provide DCFM behavior and performance characteristics during the development process

- Developers were required to provide impacts to airworthiness from the process, though the DCFM was developed specifically for laboratory use with no airworthiness requirements.

Two awards were made as a result of the BAA: Honeywell Aerospace and a Sikorsky Aircraft Corporation/The Boeing Company partnership. Each award resulted in a Technology Investment Agreement with the U.S. Army.

D. Development

1. Honeywell

The Honeywell effort delivered three components: the DCFM UoP, the RVC UoP, and an RVC Controller, as shown in Figure 8. The DCFM functionality was based on reusing an algorithm previously developed for NASA that was auto generated from Matlab models. The DCFM interface was generated from the FACE Ecosystem. The RVC was developed as a FACE UoP that would interact with an RVC Controller that runs on a test platform (such as a laptop). The RVC interface to the DCFM mimicked that of the Situational Awareness Database Manager (SADM) component from MIS. The interface between the RVC UoP and RVC Controller was constructed as an internal interface for simplicity which does not align to the FACE standard. This interface requires certain Input/Output (I/O) be available for any OE in order to execute the RVC.

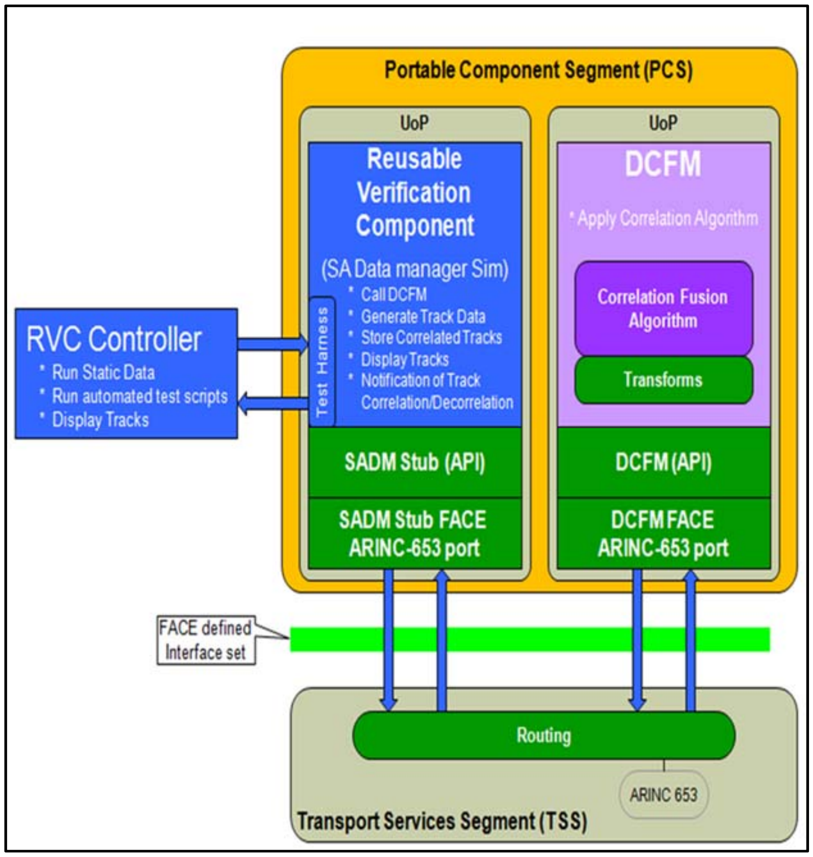

Portable Component Segment (PCS)
UoP
Reusable Verification Component
(SA Data manager Sim)
Call DCFM
Generate Track Data
Store Correlated Tracks
Display Tracks
Notification of Track Correlation/Decorrelation
Test Harness
UoP
DCFM
Apply Correlation Algorithm
Correlation Fusion Algorithm
Transforms
RVC Controller
Run Static Data
Run automated test scripts
Display Tracks
SADM Stub (API)
SADM Stub FACE ARINC-653 port
DCFM (API)
DCFM FACE ARINC-653 port
FACE defined Interface set
Routing
ARINC 653
Transport Services Segment (TSS)

Figure 8. Honeywell Development Effort

The Honeywell Final Report [28] captures the full description of the effort to include the methods and procedures used, results, conclusions, and recommendations.

2. Sikorsky/Boeing Effort

Sikorsky and Boeing are teamed on the effort to develop and demonstrate the Defiant™ aircraft as part of the JMR Air Vehicle Demonstrator (AVD) and utilized the same approach for JCA Demo. Boeing's Cohesion software provided the functionality for the DCFM UoP, as shown in Figure 9. Cohesion is a fusion application used on several military aircraft, including P-8A, Airborne Warning and Control System (AWACS), and B-1B. A software interface layer that conforms to the FACE standard and provided DCFM data model entities was added to Cohesion. The FACE Ecosystem tools produced C++ structures from the data model. The RVC was also developed as a FACE UoP, and its interface was generated from the FACE Ecosystem tools using the DCFM data model. The Sikorsky AnyCASE™ tool was used to auto-generate additional software for the RVC to implement the test cases. Domain-specific tests were generated from a modeling capability built into the AnyCASE™ tool. An unanticipated finding was that this process revealed the potential to create a generally usable RVC that does not depend on the UoP it is testing. The same meta-model used by the AnyCASE™ tool can be used for any RVC.

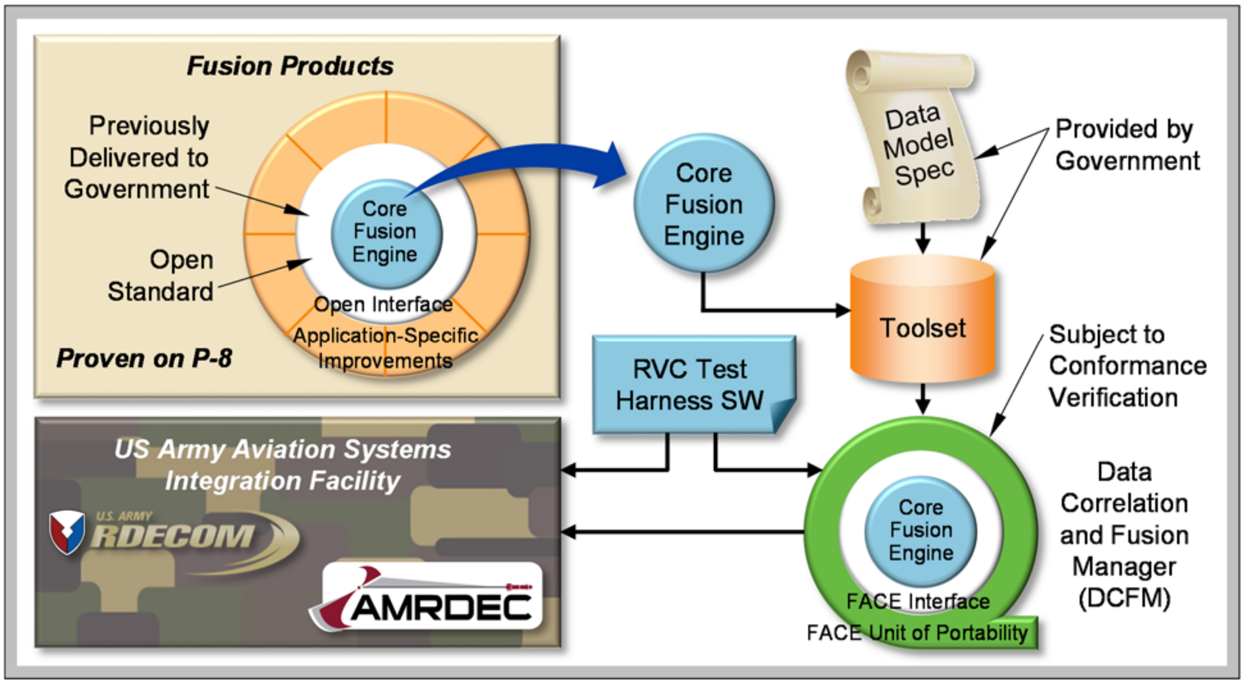


Figure 9. Sikorsky/Boeing Basic Technical Approach

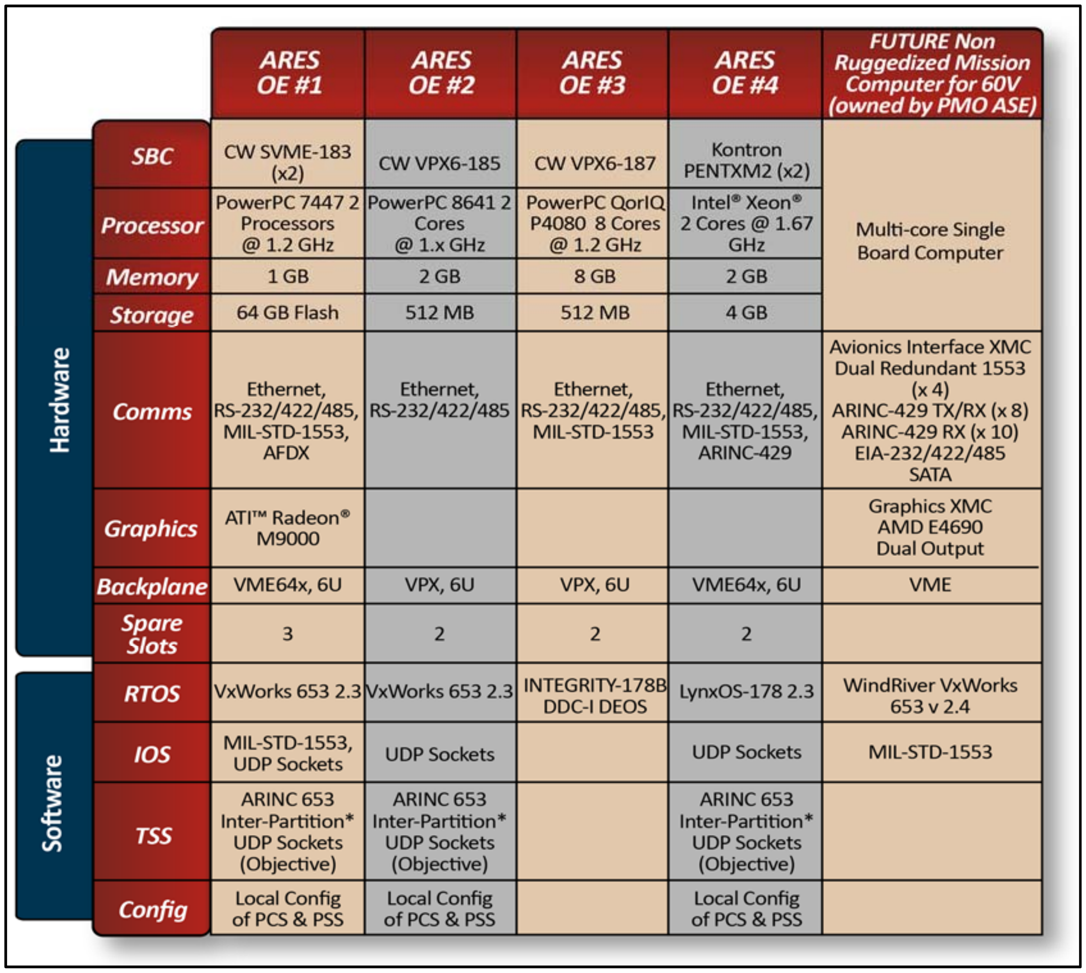
Sikorsky/Boeing performed two additional tasks beyond the BAA requirements. The first task integrated the DCFM on three different OEs representing current, emerging, and future platforms, which reinforced the findings from the government integration effort. The second task developed a flight control application as a FACE software component and measured latency introduced by adhering to the FACE architecture, which was measured at 22.6 microseconds. The Sikorsky/Boeing Final Report [29] includes a full description of the technical work accomplished, methods, and procedures used, results, conclusions, and recommendations.

E. Integration

1. System Description

The MIS S&T program created an avionics environment laboratory asset utilizing MBE and the FACE standard, which has application beyond its original intent. Now known as the AMRDEC Avionics Reference Embedded System (ARES), it comprises multiple OEs and an integrated MBE tool chain. ARES OEs 1-4, as shown in Table 2, were selected for their differences in a Processor, Real-Time Operating System (RTOS) and I/O.

Table 2. ARES OE Specifications



		ARES OE #1	ARES OE #2	ARES OE #3	ARES OE #4	FUTURE Non Ruggedized Mission Computer for 60V (owned by PMO ASE)
Hardware	SBC	CW SVME-183 (x2)	CW VPX6-185	CW VPX6-187	Kontron PENTXM2 (x2)	Multi-core Single Board Computer
	Processor	PowerPC 7447 2 Processors @ 1.2 GHz	PowerPC 8641 2 Cores @ 1.x GHz	PowerPC QorIQ P4080 8 Cores @ 1.2 GHz	Intel® Xeon® 2 Cores @ 1.67 GHz	
	Memory	1 GB	2 GB	8 GB	2 GB	
	Storage	64 GB Flash	512 MB	512 MB	4 GB	
	Comms	Ethernet, RS-232/422/485, MIL-STD-1553, AFDX	Ethernet, RS-232/422/485	Ethernet, RS-232/422/485, MIL-STD-1553	Ethernet, RS-232/422/485, MIL-STD-1553, ARINC-429	Avionics Interface XMC Dual Redundant 1553 (x 4) ARINC-429 TX/RX (x 8) ARINC-429 RX (x 10) EIA-232/422/485 SATA
	Graphics	ATI™ Radeon® M9000				Graphics XMC AMD E4690 Dual Output
	Backplane	VME64x, 6U	VPX, 6U	VPX, 6U	VME64x, 6U	VME
	Spare Slots	3	2	2	2	
Software	RTOS	VxWorks 653 2.3	VxWorks 653 2.3	INTEGRITY-178B DDC-I DEOS	LynxOS-178 2.3	WindRiver VxWorks 653 v 2.4
	IOS	MIL-STD-1553, UDP Sockets	UDP Sockets		UDP Sockets	MIL-STD-1553
	TSS	ARINC 653 Inter-Partition* UDP Sockets (Objective)	ARINC 653 Inter-Partition* UDP Sockets (Objective)		ARINC 653 Inter-Partition* UDP Sockets (Objective)	
	Config	Local Config of PCS & PSS	Local Config of PCS & PSS		Local Config of PCS & PSS	

MIS contains several representative mission system software components which were used in JCA Demo to stimulate the DCFM. The SADM gathers data from an Embedded Global Positioning System and Inertial Navigation System (EGI) Controller, APR-39D Radar Warning Receiver Controller, Weapons Watch Controller, and Tactical Data Modem Controller. These controllers are part of the FACE Platform Specific Services Segment (PSSS), which manages data and translation between the LRU specific Interface Control Documents (ICDs) and the FACE data model, enabling abstraction of the data provided to the SADM from the LRU that provides the data. The SADM interacts with the DCFM and provides results to a Display Controller for presenting the data on a digital map. While several of the LRUs were available in the ASIF, for the purposes of the demonstration, the inputs were simulated. The majority of these came from Virtual Battlespace (VBS) 3.0, which provided the virtual world and data from the interactions therein, such as the as the simulated EGI data and messages for incoming fires (APR39 and Weapon Watch). A simulated Integrated Data Modem (IDM) provided the tactical data. Figure 10 depicts the MIS system, and Figure 11 shows the architecture for JCA Demo.

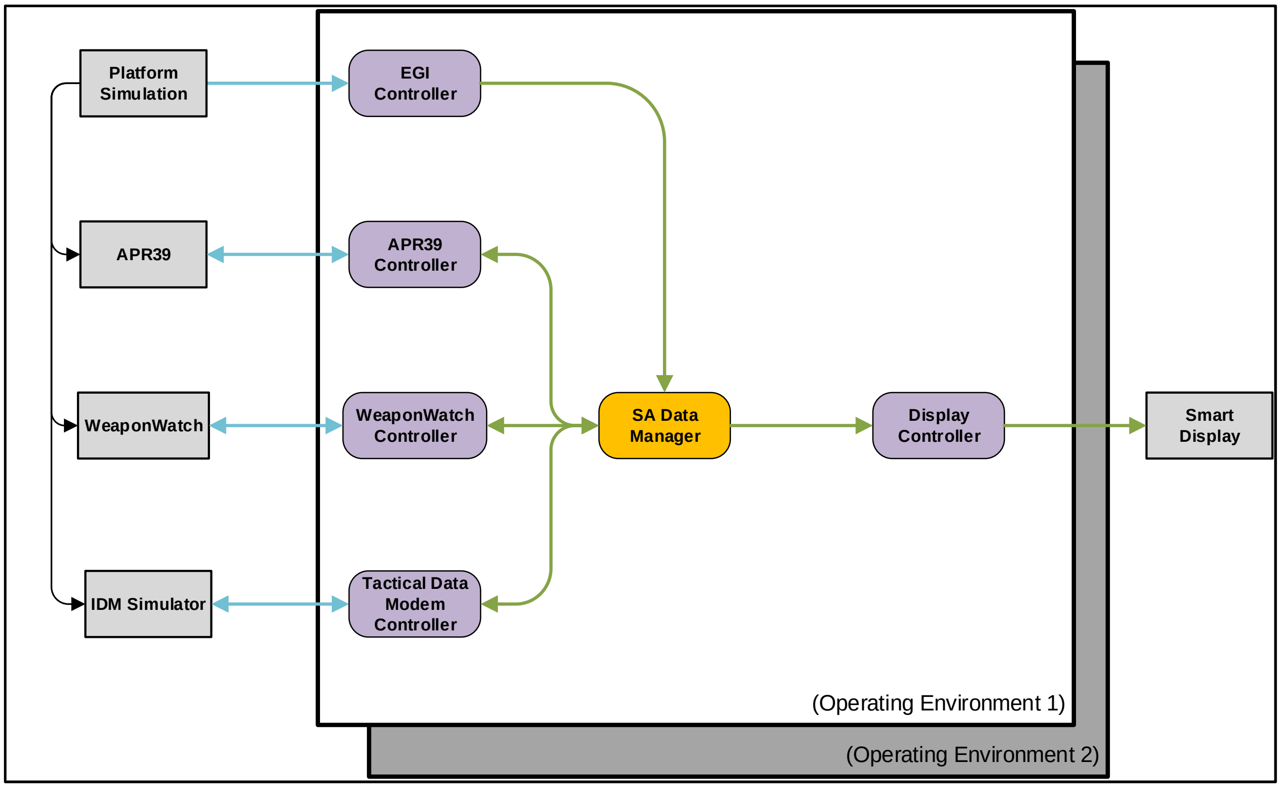


Figure 10. MIS System

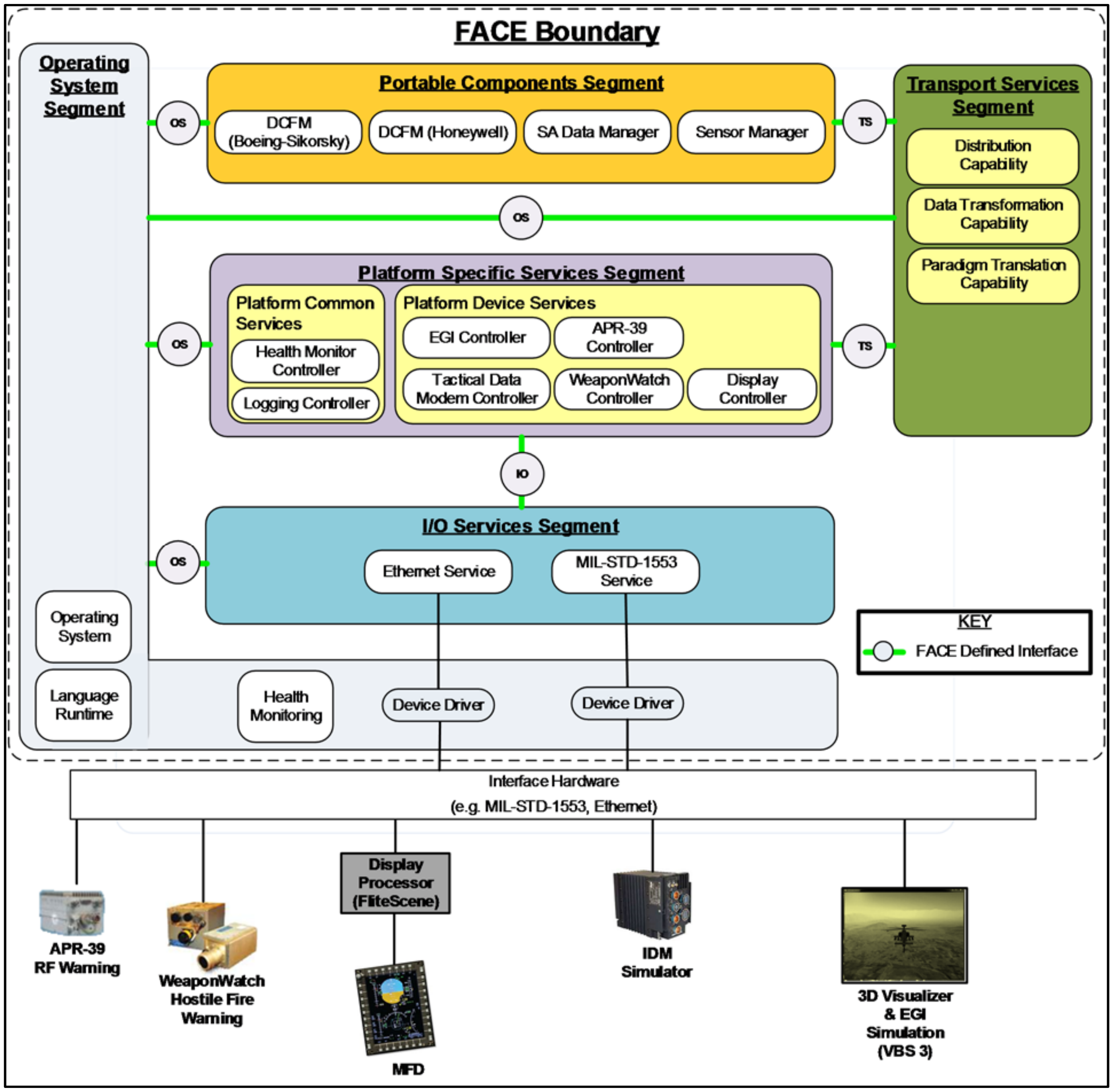


Figure 11. JCA Demo Architecture

2. Integration Approach Overview

Each software package was inspected for completeness and correct documentation when received from the DCFM developers. Following an inspection, the DCFM and RVC from each vendor was executed within a simulated Aeronautical Radio, Incorporated (ARINC) 653 environment to validate their operation against the component supplier test cases in the form of an RVC. After validation in the simulated environment, the source code was ported to the two selected OEs and integrated with the MIS system. The integration team generated test cases based on the DCFM system-level requirements. Each DCFM was tested on each OE against the test cases. Issues, concerns, corrective actions, and resolutions were documented throughout the process. Finally, the integration team prepared a laboratory demonstration scenario to display the functionality of the DCFM component in operational use. The video output was recorded for each configuration and compared to a control case (with no DCFM executing). The next three sections provide greater detail on the MIS integration activities.

In addition to the MIS integration, the Sikorsky/Boeing team successfully integrated [29] their DCFM, as shown by the green boxes in Figure 12. These OEs represent an AH-64E Apache Mission Computer, S-97 Raider Mission Computer, and an Advanced Architecture Mission Computer. Those efforts are documented in the Sikorsky/Boeing Final Report.

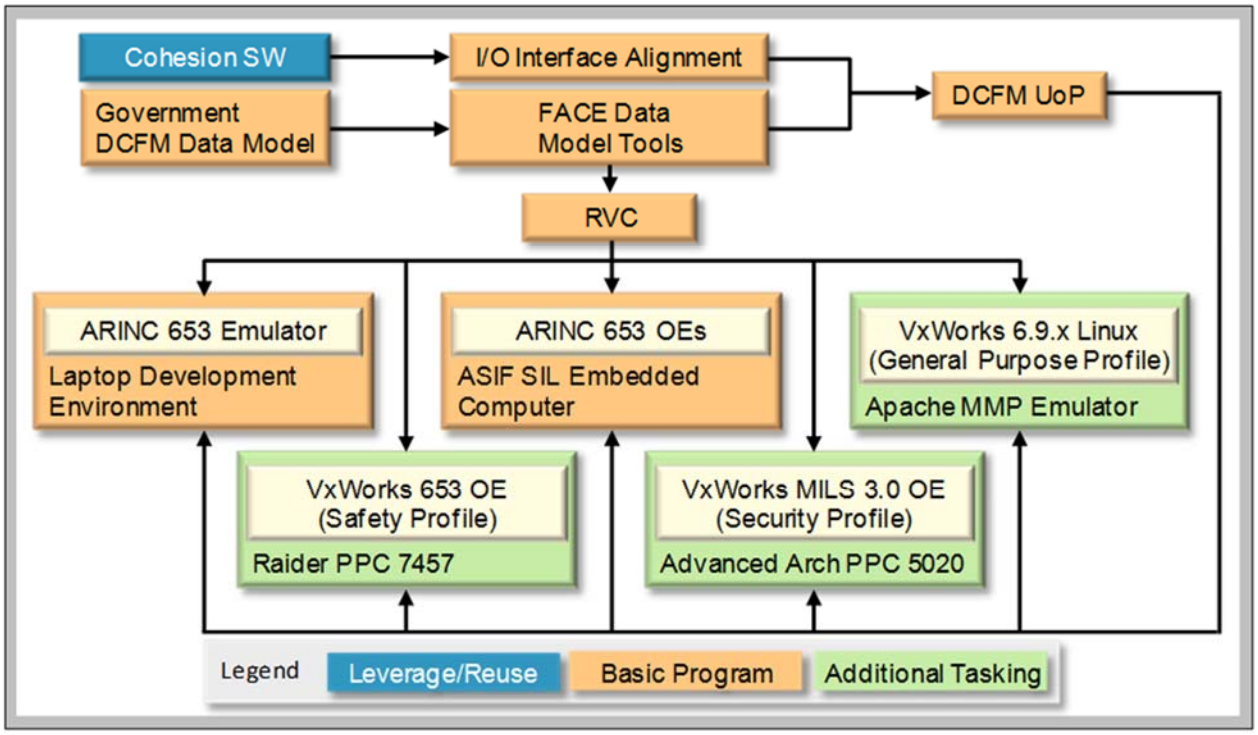


Figure 12. Sikorsky/Boeing Additional Integration Efforts

3. Integration on the ARINC 653 Emulator

The FACE tools include a simulated ARINC 653 environment that provided the method used during JCA Demo for performing early assessment on the quality of the delivered software prior to integration on the target environments. The MIS team had compilation difficulties with each DCFM for various issues ranging from details on compiler versions, build management settings in CMake, and versions of the standard C++ libraries. Identifying and resolving these issues early in the integration timeline led to significant improvements to the remainder of the integration tasks.

After successful compilation of each DCFM for the emulator, the MIS team compiled its corresponding RVC. The DCFM and RVC were executed on the emulator to verify correct operation of the delivered software against its software requirements prior to integration on the target environments.

4. Integration on OE 1

Integration on OE 1 encountered minor issues due to a lack of RTOS conformance to the FACE Operating System Segment (OSS) and ARINC 653 standardization of certain elements. These issues were fixed by generating a wrapper to provide an OS interface aligned to the FACE standard along with minor software modifications. Table 3 captures the issues, their resolutions, and recommendations going forward.

Table 3. Integration Issues on OE 1

Issue	Resolution	Recommendation
Difference in header file naming convention between RTOS	Modified delivered software to match header files provided by RTOS	ARINC 653 should standardize header file naming conventions
VxWorks 653 configuration data requirements varied from the ARINC 653 Emulator	Changed configuration data parameters	ARINC 653 should standardize configuration data requirements
VxWorks653 configuration limited connection names for queueing ports to 30 characters	Shortened connection names	ARINC 653 should standardize name lengths
VxWorks653 operating system scheduler did not correctly execute periodic activities that worked on the simulated ARINC-653 environment	Added loop to ensure that core processing occurred periodically	
The FACE tools generated function calls specific to the ARINC 653 Emulator	Manually replaced with VxWorks 653 function calls	FACE tools should generate generic code that only depends on RTOS APIs
Resolution of issues	Integration Team modified DCFM source code to resolve issues previously listed	Transport Services (TS) implementation should have been modified to resolve integration issues

5. Integration on OE 2

Integration on OE 2 also encountered issues due to the lack of RTOS conformance to the FACE standard. Additional issues were observed such as lack of LynxOS-178 compiler support and process execution that was noncompliant to the ARINC 653 standard, which could cause problems when reusing object code. These issues were fixed by generating a wrapper to provide an OS interface aligned to the FACE standard along with minor software modification changes. Table 4 captures those issues, their resolutions, and recommendations going forward.

Table 4. Integration Issues on OE 2

Issue	Resolution	Recommendation
Difference in header file naming convention between RTOS	Modified delivered software to match header files provided by RTOS	ARINC 653 should standardize header file naming conventions
LynxOS-178 did not support < and > characters used for port names	Changed port name characters	ARINC 653 should standardize allowable characters
LynxOS-178 requires ARINC 653 processes as separate executables	Converted to Portable Operating System Interface for Unix (POSIX) threads	LynxOS-178 compliance to ARINC 653 standard
LynxOS-178 lack of compiler support	Added parentheses to enforce proper order of operations in matrix mathematics for several files that would not compile due to incorrect interpretation	LynxOS-178 Standardize compiler support
LynxOS-178 lack of compiler support	Changed and to &&	LynxOS-178 Standardize compiler support
LynxOS-178 lack of compiler support	Replaced std::numeric_limits<unsigned long>::max()” with ULONG_MAX	LynxOS-178 Standardize compiler support
LynxOS-178 lack of compiler support	Added #include <new> statement to get a “placement new” type of new operator to compile	LynxOS-178 Standardize compiler support
Resolution of issues	Integration Team modified DCFM source code to resolve issues previously listed	TS implementation should have been modified to resolve integration issues

F. Verification

The delivered software was verified for performance against the DCFM requirements and conformance to the FACE Technical Standard. Performance requirements were verified at the unit and system levels. Conformance to the FACE Technical Standard occurred through the Army's FACE VA.

1. Unit Level Test

Each vendor was required to deliver an RVC to test their DCFM against its software requirements. The RVCs were to ensure that unit level tests could be performed on any potential OE. The approach to verification was to first ensure correct operation of the delivered DCFM prior to integration on the OEs. This occurred utilizing the ARINC 653 Emulator included in the FACE tool suite. The next logical step would be to integrate the DCFM and RVC on the OE to perform unit level test; however, this was not included in the program schedule and did not occur until several months after completion of the demonstration. The RVC approach showed potential to satisfy some airworthiness requirements for RSCs [30].

a. Honeywell RVC

The first verification of the Honeywell DCFM using their RVC occurred on the ARINC 653 Emulator running on Linux. The only modification necessary was due to the 64-bit Linux where long integers were defaulted to 8 bytes, which had to be changed back to 4 bytes. After correction, the Honeywell DCFM passed all tests executed by the RVC.

Integrating the Honeywell RVC on OE 1 required significant effort. Since the Honeywell RVC utilized an external controller application to communicate with the RVC UoP running on the OE, the RVC UoP was dependent on the availability of the User Datagram Protocol (UDP) running over an Ethernet interface. For OE 1, the WindRiver VxWorks 653 RTOS does not provide this interface directly to applications, which required the MIS team to redesign Honeywell's RVC UoP for this OE, as shown in Figure 13. Furthermore the Honeywell RVC UoP directly called POSIX Sockets from a partition which is not permitted in the FACE standard or supported by VxWorks 653. The MIS team replaced the UDP Ethernet interface with a FACE aligned Input/Output Segment (IOS) interface to achieve UDP communications. The MIS team also had to resolve an issue with endianness (byte ordering) between the RVC UoP and the external controller. Finally, some minor changes were required due to the use of C++ objects and functions not allowed by the FACE standard or supported by the RTOS. For OE 2, similar issues to OE 1 were required, including replacing the UDP interface, correcting for endianness, and the C++ functionality. The Honeywell DCFM ultimately passed all tests executed by the RVC on OE 1 and OE 2.

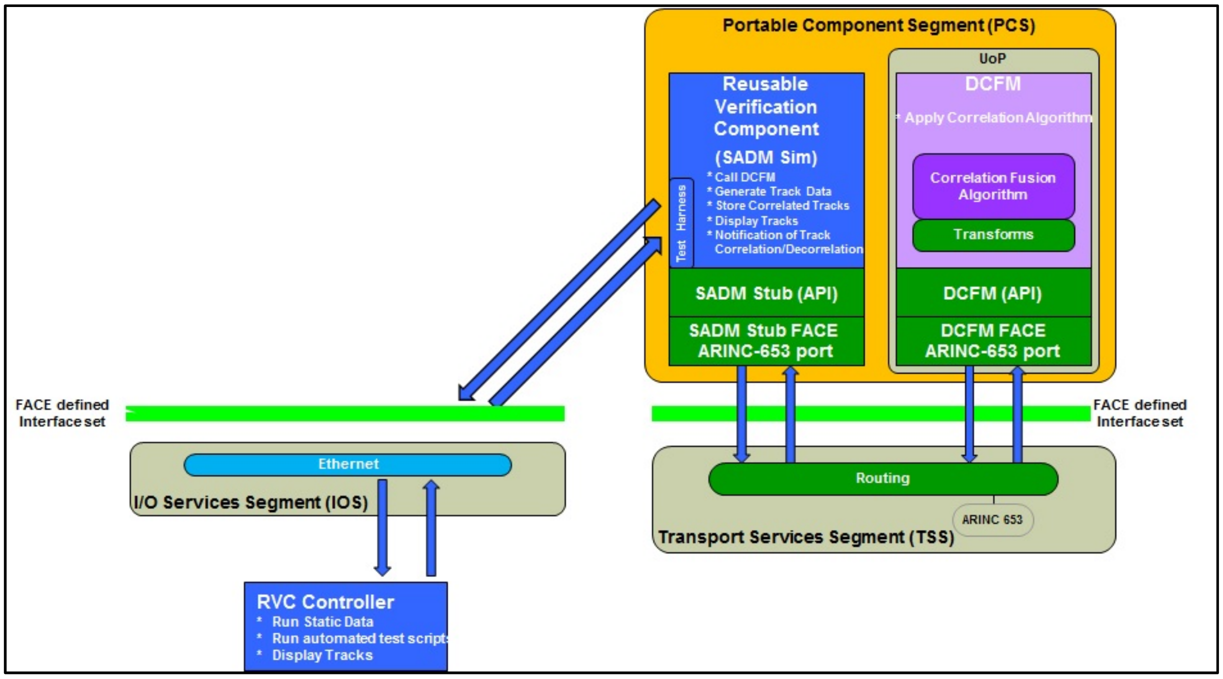

Portable Component Segment (PCS)
UoP
Reusable Verification Component
(SADM Sim)
* Call DCFM
* Generate Track Data
* Store Correlated Tracks
* Display Tracks
* Notification of Track Correlation/Decorrelation
Test Harness
DCFM
* Apply Correlation Algorithm
Correlation Fusion Algorithm
Transforms
SADM Stub (API)
SADM Stub FACE ARINC-653 port
DCFM (API)
DCFM FACE ARINC-653 port
FACE defined Interface set
FACE defined Interface set
Ethernet
I/O Services Segment (IOS)
Routing
ARINC 653
Transport Services Segment (TSS)
RVC Controller
* Run Static Data
* Run automated test scripts
* Display Tracks

Figure 13. Redesign of Honeywell Reusable Verification Component

b. Sikorsky/Boeing RVC

The first verification of the Sikorsky/Boeing DCFM using their RVC occurred on the ARINC 653 Emulator running on Linux. The software components were compiled and executed with little effort. The DCFM passed all tests of the RVC. Since the Sikorsky/Boeing RVC was developed completely as a FACE application, integration onto each OE was straightforward. The changes necessary to the RVC mirrored the ones observed when integrating the DCFM on the OE and are described in Tables 2 and 3. The Sikorsky/Boeing DCFM passed all tests executed by the RVC on OE 1.

c. Mixing RVCs Between DCFMs

During execution of JCA Demo it was frequently asked “what would happen if you ran the RVC from one vendor against the other vendor’s DCFM?” After completing the RVC efforts for OE 1, the MIS team attempted it. The Sikorsky/Boeing RVC successfully executed against the Honeywell DCFM on OE 1 with nearly identical test results (compared to the Sikorsky/Boeing DCFM). This was not the case when testing the Sikorsky/Boeing DCFM with the Honeywell RVC.

While JCA Demo only contained minimal high-level requirements for basic functionality, both the Honeywell and Sikorsky/Boeing DCFMs were derived from existing software components that had been previously defined by their own set of functional and performance requirements. Those requirements were implemented within a specific architecture and designed for a specific environment or set of environments. Even if the high-level requirements were identical to those in JCA Demo, a deviation between low-level requirements resulted in mismatches that were not initially recognized and required specific integration activities to resolve.

For example, a discrepancy occurred over the necessity of certain data fields in the set of test data. This was uncovered when swapping the RVCs. The Sikorsky/Boeing DCFM required an error association greater than zero to determine validity while the Honeywell DCFM was ambivalent to the data value. As a result, when the Honeywell RVC provided source tracks with location errors that equaled zero, the Sikorsky/Boeing DCFM did not include them for correlation. This represents a mismatch of low-level requirements that would require additional integration work or modification of the original software to meet the requirements.

Another issue occurred regarding how each DCFM treated data coming from sensors. The Sikorsky/Boeing DCFM does not correlate tracks that come from the same source (for example, multiple tracks from a Radar Warning Receiver). This low-level requirement represents a design decision based upon system characteristics, in this case, that all sensors have predetermined their outputs to be unique tracks. The failure of the Sikorsky/Boeing DCFM to perform properly when tested by the Honeywell RVC is a good illustration of issues that can be encountered when trying to reuse a component in systems that have different low-level requirements. This underscores that RSC's reflect architectural decisions that may or may not be apparent to a reusing organization. Such inherited decisions have a direct impact on the ease of reuse and suitability of the RSC and require sufficient understanding of the RSC behavior and data. In this case, the issue may have been uncovered through the normal course of requirements development, analysis, and technical reviews but only if sufficient documentation exists and it is available for the integrator to use. To ensure this, a formal software development process, such as RTCA's DO-178, and data right strategy is required.

During this effort, it was observed that the RVC can serve as an abstraction of the target system from the perspective of the DCFM. This has led to preliminary concepts of a system-level RVC that can be provided in solicitation to convey a golden test case for the target system that could serve as an early integration suitability test along the lines of ACVIP. This concept needs to be further refined and investigated in future experiments.

JCA Demo did not investigate elements of dead or deactivated code that would inherently exist within an RSC or how an RVC would aid in the demonstration of deactivation mechanisms. This is of paramount importance in the airworthiness considerations for an RSC and should be investigated in future efforts.

Mixing RVCs revealed challenges of software interchangeability that unless all high- and low-level requirements are identical between software components, there is no guarantee of correct software performance.

2. System-Level Test

The MIS team generated a System-Level Integration test to verify the system level requirements (excluding the FACE requirements). To perform this test, a scenario was created to simulated aircraft platform motion and sensor detection of threats. The sensor data was aggregated by the SADM and provided to the DCFM as input. Correlated output was collected from the DCFM by the SADM and transferred to an external map display to visually present the host aircraft location, input source tracks, and output correlated tracks. I/O data were collected for post-run analysis. The test results based on OE are in Tables 5 and 6.

Table 5. System Level Test Results on OE 1

Requirement	Method	Vendor	Result
The DCFM shall analyze uncorrelated source tracks in order to identify a single correlated track believed to represent the same uncorrelated source tracks, combining the data from the duplicate uncorrelated source tracks, as appropriate	Demonstration	Honeywell	Pass
		Sikorsky Boeing	Pass
The DCFM shall analyze correlated source tracks in order to identify separate tracks, breaking the linkage of the correlated track with the uncorrelated source tracks, as appropriate	Test	Honeywell	Pass
		Sikorsky Boeing	**Fail**
The DCFM shall analyze tracks within 25 km of own-ship position	Demonstration Test	Honeywell	Pass
		Sikorsky Boeing	Pass
The DCFM shall correlate or decorrelate 50 source tracks within 1 second	Test/Inspection	Honeywell	**Fail**
		Sikorsky Boeing	Pass

Table 6. System Level Test Results on OE 2

Requirement	Method	Vendor	Result
The DCFM shall analyze uncorrelated source tracks in order to identify a single correlated track believed to represent the same uncorrelated source tracks, combining the data from the duplicate uncorrelated source tracks, as appropriate	Demonstration	Honeywell	Pass
		Sikorsky Boeing	Pass
The DCFM shall analyze correlated source tracks in order to identify separate tracks, breaking the linkage of the correlated track with the uncorrelated source tracks, as appropriate	Test	Honeywell	Pass
		Sikorsky Boeing	**Fail**
The DCFM shall analyze tracks within 25 km of own-ship position	Demonstration Test	Honeywell	Pass
		Sikorsky Boeing	Pass
The DCFM shall correlate or decorrelate 50 source tracks within 1 second	Test/Inspection	Honeywell	Pass
		Sikorsky Boeing	Pass

The Honeywell DCFM failure to correlate within the required 1 second on OE1 is attributed to the modification of the VxWorks 653 scheduling which negatively impacted performance. This issue was not observed on OE 2 as the scheduling was not modified.

The Sikorsky/Boeing DCFM failed to decorrelate tracks within the parameters of the test for both OE 1 and OE 2. MIS analysis identified that the parameters used for the system test were insufficient to trigger a decorrelation event for their algorithm. Sikorsky/Boeing developers confirmed the MIS analysis and verified that the DCFM was operating in accordance with its software requirements. This disconnect between system-level and software requirements is another example of a barrier to software reuse, requiring Systems Integrators to analyze all software requirements for second- and third-order effects to system level tests when reused on a new system. One proposed approach to reduce the likelihood of this disconnect would be to include a system level test data set with the solicitation to enable third-party developers to understand the interaction and implications for their component.

3. FACE Verification

The Army's FACE VA was utilized to verify that each vendor's DCFM met the FACE requirements. The BAA required delivery of preliminary, initial, and final FACE verification packages. As previously mentioned, it was known that the DCFM data model would not pass conformance to the FACE SDM.

FACE Conformance verification has three major elements:

1) Conformance Test Suite (CTS)

2) Conformance Matrix

3) Conformance Statement

The CTS is a FACE Ecosystem computing tool that analyzes the post-compilation object code of the candidate UoP to ensure that the software uses ARINC 653 or POSIX calls allowed by the FACE standard for the selected profile for that operating system. The tool produces pass or fail results. The conformance matrix is a list of requirements as specified by the FACE standard for visual inspection. It identifies requirements for additional documentation that need to be provided to show that the software conforms to the standard. The VA checks the submitted matrix and documentation for correctness and completeness. The conformance statement is prepared by the developer to identify UoP details, such as version number and selected operating system profile. It may also list conditional requirements that were not addressed.

Some of the issues encountered during the VA process stemmed from the lack of availability of the FACE Conformance Guide, which had been developed but not published by the FACE Consortium. A draft version of this guide was delivered to JCA Demo participants after the problems were discovered. The Army's FACE VA also generated a guide to their processes as a result of these issues and updated their process documents to address the lessons learned.

The process for FACE verification, certification and registration is shown in Figure 14. JCA Demo interaction with the FACE VA included submission for certification, even though the component was known not to be conformant. This was done to exercise the process and determine if there were any issues.

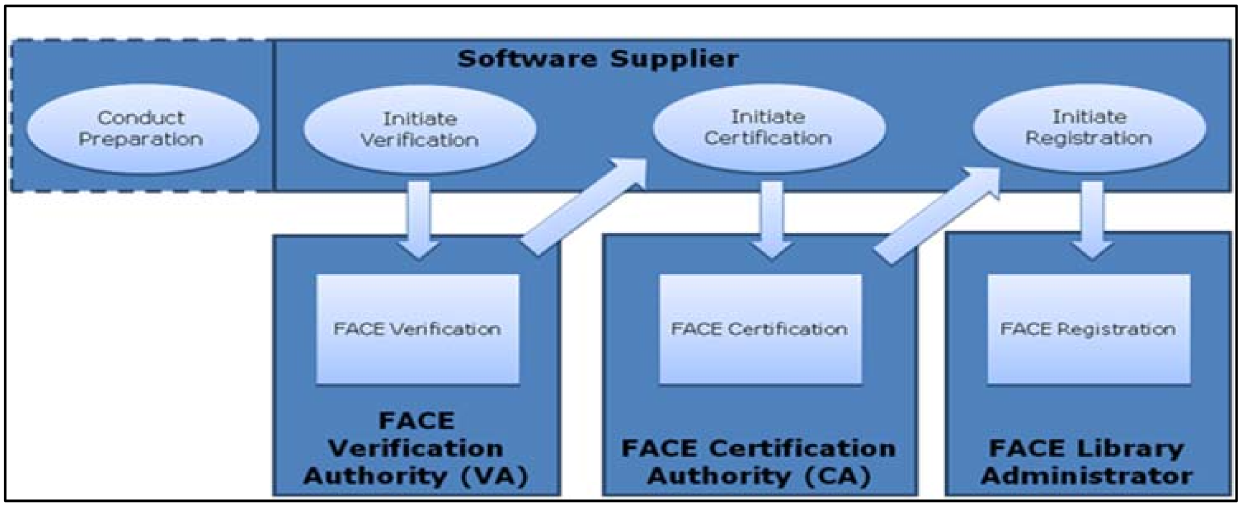


Figure 14. FACE Verification, Certification and Registration Process

It was decided to handle the verification and mock certification process differently between the Sikorsky/Boeing team and the Honeywell team. Sikorsky/Boeing, as the DCFM Supplier, submitted their FACE CTS verification data and Conformance Statement directly to the FACE VA. Honeywell, as the other DCFM Supplier, submitted their FACE CTS verification data and Conformance Statement to the government project engineer, who then coordinated with the FACE VA as a government Product Management Office (PMO) surrogate to demonstrate another method of verification.

a. Honeywell DCFM FACE Verification

For the Honeywell effort, the government team served as the submitting PMO organization to the VA based on the deliverables from the vendor. This was done to see how the government would interact with the FACE VA in following the process for obtaining certification. Honeywell executed the FACE CTS before the delivery of the software and submitted the results along with the necessary technical artifacts (for example, requirements, software design documents, verification documents, and so forth) to the government PMO surrogate. The government PMO surrogate submitted the CTS results to the FACE VA, who requested that a Conformance Statement be submitted in conjunction with the CTS results and artifacts. The Conformance Statement was missing instructions and vague in areas and therefore, required clarification by the FACE VA. This resulted in generation of a Technical Instruction outlining the process for the verification and registration process, plus improvement of the Conformance Statement instructions. These instructions were coordinated with Honeywell who provided the answers to complete the submission of the DCFM's CTS results, Conformance Statement, artifacts, and software to the government PMO to review and relay to the FACE VA. Also as part of this process, the software was registered on the FACE Registration Site. The FACE Library Registry website contains questions and requires data be submitted. However, the instructions were vague so the government PMO surrogate coordinated with the FACE Library Registry website administrator to correct the shortcomings in the instructions. The government PMO surrogate completed the registration process with the verification submission on the FACE Registration website. Exercising this process was very beneficial in improving the FACE VA instructions and FACE Library Registry website instructions.

During post-software development, the DCFM software and data model were run through the FACE CTS. The Honeywell DCFM Component passed verification testing, except for conformance to the FACE SDM 2.0, as expected.

b. Sikorsky/Boeing DCFM FACE Verification

Sikorsky/Boeing worked directly with the VA to submit documentation and resolve issues. This included utilizing the beta version of the FACE Library Registry, particularly the areas meant to guide developers through the verification process. Sikorsky/Boeing found similar issues with the FACE Registry process and Conformance Statement that were resolved.

In the preliminary submission, the VA found unallowable POSIX calls that were in the original Cohesion source code, which were resolved with conditional compiler flags. During the initial submission, the VA identified a false positive that was corrected by changing the input to the conformance suite and adding compiler-specific code. The final submission passed all requirements except for the expected conformance issue associated with the FACE SDM 2.0.

G. Demonstration

The purpose of the demonstration was to show portability of the independently developed software components correctly performing DCFM functionality on both OEs. A scenario was created that would depict track data on a moving map that could be recorded in order to present visual evidence of DCFM operation.

1. Demonstration Scenario

A single correlation event scenario was used for the demonstration. In the scenario, an Unmanned Aircraft System (UAS) is tasked to locate insurgent activity. It detects and reports the location of a possible threat to a simulated helicopter through a tactical network as a Variable Message Format (VMF) message, which is provided to the SADM. The helicopter navigates to the location and uses onboard sensors to detect hostile fire from the previously identified threat and provides the information to the SADM. The DCFM then correlates the twice reported threat into a single entity. The helicopter engages the hostile threat to complete the mission.

2. Demonstration Infrastructure

In addition to the two OEs, the following infrastructure was used for the demonstration. VBS 3.0 was used to model the scenario used for demonstration. Harris

FliteScene™ was used to display output from the representative mission system on a map. MIS Threat Stimulator software application converted messages from VBS into representative formats for each LRU.

3. Demonstration Execution

The scenario was executed for all OE/DCFM configurations to include a control (no DCFM). Video was captured of the visualization from VBS and FliteScene™ for each DCFM/OE configuration. For each OE, a video was stitched together, as shown in Figure 15. Video displaying the VBS 3.0 view of the scenario (top left), baseline configuration (bottom left), the Sikorsky/Boeing DCFM (top right), and the Honeywell DCFM (bottom right) were all displayed on a video screen at the same time. The captured videos showed both DCFMs functioned correctly on both OEs.

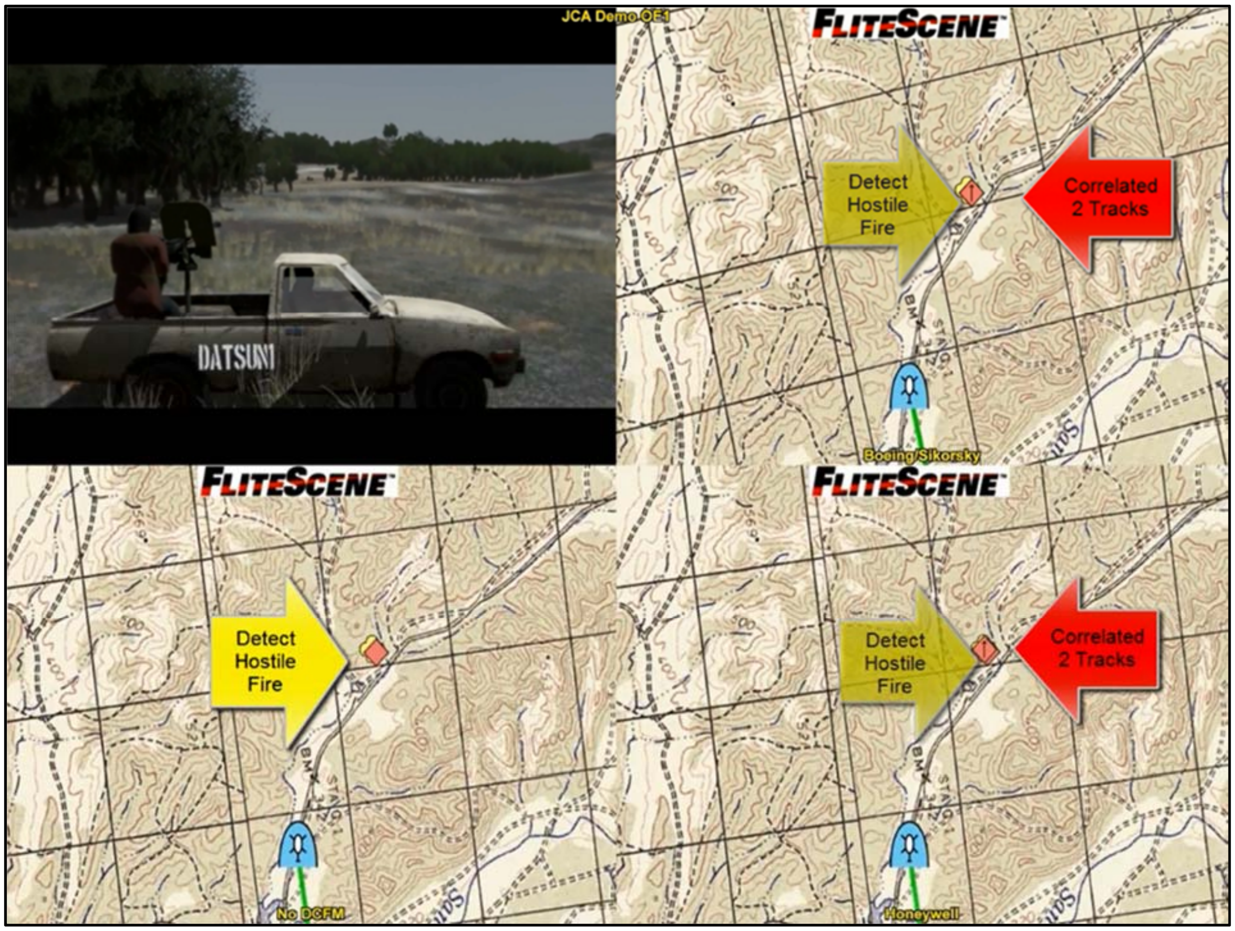


Figure 15. Demonstration Scenario

IV. CONCLUSIONS AND RECOMMENDATIONS

JCA Demo was designed to investigate the potential of using OSA, FRA, and MBE in an acquisition approach to efficiently achieve strategic software reuse. The results show that it was possible in a narrowly defined case involving very few components with minimal interactions and relatively benign performance requirements. The approach taken by JCA Demo to use the government as the systems integrator and tightly regulate interaction between the component developers and integrators proved effective in generating significant lessons learned related to the processes, tools, and standards employed. Considerable additional investigation is required before the government can be confident that the processes, tools, and standards are sufficiently capable and mature enough to address the challenges presented by the complexity of modern aircraft systems. The following sections contain details pertaining to specific aspects of the processes, standards, and tools used during JCA Demo.

A. Future Airborne Capability Environment Technical Standard

The FACE Technical Standard provided the common software environment that enabled software portability across OEs. This portability is critical for efficient software reuse but does not address the functionality and behavior of the component or the full environment into which the component will be integrated.

The government, acting in the role of Systems Integrator, provided only minimal behavioral and performance information to the vendors regarding the interaction of the defined interface. Through utilization of the FACE standard, vendors were able to deliver software components that could be integrated onto each OE despite not knowing which processors, RTOSs, or transport mechanisms that the software would execute on or the types and combinations of sensors from which it would receive data.

While the FACE data model provided symantic and syntactic understanding of the interface data elements, it did not include behavioral information on the system. As a result, the vendors were required to make assumptions on the architecture and design of the target system. These assumptions were not stressed under JCA Demo but could prove problematic under a more stringent set of system requirements and test cases thus requiring more than a data model.

Training on the FACE Technical Standard was observed to be a challenge for personnel not involved in the consortium. Formal training and best practices do not currently exist for the FACE standard. This shortfall is exacerbated by the requirement for data modeling that requires an approach to interface development that may not be familiar to all parties involved. Creating the DCFM data model in accordance with the FACE data architecture was labor intensive. It is unclear if this was due to pathfinding the new process or whether the model was efficiently scaled for complex components.

Vendors noted that it was relatively easy to convert their modular, legacy software code to a FACE UoP. It is unknown what level of effort would have been required to develop the DCFM from scratch as a FACE UoP. Future experimentation should include new software development alongside the conversion of legacy software to a FACE UoP to enable comparisons of the approaches and to identify impacts and implications.

Key elements of the FACE Conformance process were executed, matured, and found to be ready for use. No fundamental flaws were found in the FACE standard. Future efforts should maintain close coordination with the FACE Consortium and utilize the change processes to ensure that future enhancements to the FACE standard and Ecosystem occur in a timely manner.

B. Joint Common Architecture Functional Reference Architecture and Model-Based Engineering

The JCA FRA, combined with a series of high-level sequence diagrams and textual performance requirements, constituted the supporting data model for specification of the DCFMs. Modularity was achieved through the functionality constrained by JCA, model-based interface, and textual requirements specification. Improvements were made to the JCA functional and data models as well as the development process for future efforts. Challenges were uncovered over the need for behavioral modeling, the interaction between data and behavioral models, methods to convey critical architectural context, and the level of requirements specificity necessary to achieve interchangeability. Future demonstrations should try to determine the minimum amount of behavioral specification necessary (sequence, activity, state, use cases, and so forth) to describe required component/system interaction to efficiently achieve reuse.

Given the limited goals of the JCA Demo, it remains unclear how the necessary models will be developed in an FVL acquisition environment and which organizations will be responsible for developing them at each point in the process. Model-based processes are anticipated to be the primary method of acquiring software capability on future programs. The government's role involves architectural definition and specification at the conceptual and logical level. Modeling at those levels requires an understanding of the informational needs to be conveyed and a larger, more comprehensive vision for strategic software reuse. Neither the understanding nor the vision is well-defined at this point, which underscores the importance of continuing to perform demonstrations like JCA Demo. A component's functional description must be carefully coordinated with a corresponding data description so that context and consistency are maintained. Significant investment will need to be made in both tool development and training in order to provide automation and translation functions that currently are challenging and time consuming. Training is necessary to assist all participants in changing the current culture of paper-based specifications and understanding the proper way to implement a model-based development environment. The potential benefit justifies the investments in training for the FACE Data Architecture and ACVIP methods.

Utilizing a model-based specification together with JCA-defined functionality provided a limited amount of interchangeability; however, challenges were uncovered that will impact the potential future reuse of software in complex, safety critical systems. Differences in low-level requirements result in integration challenges, rework, or fielded deficiencies. Since low-level requirements play a significant role in the overall cost of major defense acquisitions [31], the impact of differences could preclude efficient software reuse.

C. Reusable Verification Component

The RVC approach showed potential to satisfy some airworthiness requirements for RSCs. Specific qualification objectives lend themselves to verification through an RVC approach to testing; however, the terms under which such practice may be considered as part of a qualification case are different on a case by case basis. Each Systems Integrator must interact with their Airworthiness Authority to achieve concurrence on a plan for qualification. This plan would necessarily include details as to how the partial qualification pedigree and/or automated testing of third-party software components might be part of the qualification case for a software application, LRU, or avionics system. The RVC was beneficial to the Systems Integrator and could provide a key element of the airworthiness qualification. Implementing the RVC as a FACE software component provided the portability necessary to execute alongside of the RSC on different OEs.

D. Experiment Approach

By conducting JCA Demo as a controlled experiment designed to learn by doing enabled findings outside of what would traditionally be observed through a demonstration. Limiting communications during integration uncovered additional findings that would have simply been resolved on the side although it impeded the integration task. By emphasizing lessons learned as the most important element of the demonstration timely changes to the JCA process, FACE Technical Standard, FACE tools, and MBE methods were possible.

In total, 49 lessons learned were captured during the execution of JCA Demo. The complete set of lessons learned and recommendations can be found in Appendix C.

E. Next Steps

It is recommended that MSAD continue with its plan of performing increasing complex, robust experiments and demonstrations utilizing representative FVL acquisition approaches. The breadth, magnitude, and ability to action the lessons learned from the JCA Demo validated the technical approach of learning by doing and resulted in the maturation of the JCA process, FACE Technical Standard, FACE tools, MBE, and ACVIP methods to address FVL FOS complexity and affordability.

It would be beneficial for these standards, processes, and tools to also be matured by other organizations, preferably through demonstrations similar to JCA Demo that involve multiple systems and components from different organizations filling various roles, while controlling the interactions to truly learn by doing. These kinds of demonstrations significantly decrease risk for follow-on implementation by program offices.

REFERENCES

1. “DEFENSE ACQUISITIONS: Stronger Management Practices Are Needed to Improve DOD’s Software-Intensive Weapon Acquisitions,” GAO-04-393, Government Accountability Office, March 2004.

2. Hansen, M. and Nesbit, R., “Report of the Defense Science Board Task Force on Defense Software,” Defense Science Board, November 2000.

3. Redman, D. et al., “Virtual Integration for Improved System Design,” Analytic Virtual Integration of Cyber-Physical Systems Workshop, November 2010.

4. “LONGBOW APACHE HELICOPTER: Systems Procurement Issues Need to be Resolved,” Government Accountability Office (GAO)/NSIAD-95-159, GAO, August 1995.

5. “COMANCHE HELICOPTER: Testing Needs to be Completed Prior to Production Decision,” Government Accountability Office (GAO)/NSIAD-95-112, GAO, May 1995.

6. “EMBEDDED COMPUTER SYSTEMS: C-17 Software Development Problems,” Government Accountability Office (GAO)/IMTEC-92-48, GAO, May 1992.

7. “EMBEDDED COMPUTER SYSTEMS: New F/A-18 Capabilities Impact Navy’s Software Development Process,” Government Accountability Office (GAO)/ IMTEC-92-81, GAO, September 1992.

8. “TACTICAL AIRCRAFT: F-22A: Modernization Program Faces Cost, Technical and Sustainment Risks,” Government Accountability Office (GAO)-12-447, GAO, May 2012.

9. “JOINT STRIKE FIGHTER: Restructuring Places Program on Firmer Footing, but Progress Still Lags,” Government Accountability Office (GAO)-11-325, GAO, April 2011.

10. “F-35 SUSTAINMENT: Need for Affordable Strategy, Greater Attention to Risks, and Improved Cost Estimates,” Government Accountability Office (GAO)-14-778, GAO, September 2014.

11. “F-35 JOINT STRIKE FIGHTER: Problems Completing Software Testing May Hinder Delivery of Expected Warfighting Capabilities,” Government Accountability Office (GAO)-14-322, GAO, March 2014.

12. “DEFENSE ACQUISITIONS: Assessments of Selected Weapon Programs,” Government Accountability Office (GAO)/AO-15-342SP, GAO, March 2015.

13. Foreman, V. et al., “Software in military aviation and drone mishaps: Analysis and recommendations for the investigation process,” Journal of Reliability Engineering and System Safety, January 2015.

REFERENCES (CONTINUED)

14. Dvorac, D., “NASA Study on Flight Software Complexity,” National Aeronautics and Space Administration (NASA), Office of Chief Engineer, 2009.

15. Dova, V., “Software-Defined Avionics and Mission Systems in Future Vertical Lift Aircraft,” National Aeronautics and Space Administration (NASA), Postgraduate School, March 2015.

16. “DoD Open Systems Architecture: Contract Guidebook for Program Managers,” Department of Defense (DoD) Open Systems Architecture Data Rights Team, Version 1.1, June 2013.

17. “Integrated Modular Avionics IMA Development Guidance and Certification Considerations,” DO-297, Radio Technical Commission for Aeronautics, November 2005.

18. “Reusable Software Components,” AC 20-148, Federal Aviation Administration (FAA), December 2004.

19. Feiler, P. and Gluch, D., “Model-based Engineering with AADL: An introduction to the SAE Architecture Analysis & Design Language,” Addison-Wesley, 2012.

20. Boydston, A. et al. “Joint Common Architecture Demonstration Architecture Centric Virtual Integration Process (ACVIP) Shadow Effort,” American Helicopter Society, 71st Annual Forum, Virginia Beach, VA., May 2015.

21. Vestal, S., “Joint Common Architecture Demonstration Shadow Architecture Centric Virtual Integration Process–Final Technical Report,” Adventium Labs, October 2015.

22. Feiler, P. and Hudak, J., “Potential System Integration Issues in the JMR JCA Demonstration System,” Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA., December 2015.

23. Feiler, P. “Requirements and Architecture Specification of the JMR JCA Demonstration System,” Carnegie Mellon University (CMU)/Software Engineering Institute (SEI) 2015-SR-030, SEI, CMU, Pittsburgh, PA, December 2015.

24. Feiler, P. “Architecture Led Safety Analysis of the JMR JCA Demonstration System,” Carnegie Mellon University (CMU)/ Software Engineering Institute (SEI) 2015-SR-032, SEI, CMU, Pittsburgh, PA, November 2015.

25. Technical Standard for Future Airborne Capability Environment (FACE^{TM}) Technical Standard, Edition 2.0, The Open Group, February 2012.

REFERENCES (CONCLUDED)

26. "Developer's Handbook for Airworthy, Reusable FACE Units of Conformance," United States (U.S.) Army, Aviation and Missile Research, Development, and Engineering Center (AMRDEC), Software Engineering Directorate (SED), Redstone Arsenal, AL, April 2014.

27. "DEFENSE ACQUISITION PROCESS: Military Service Chiefs' Concerns Reflect Need to Better Define Requirements before Programs Start," Government Accountability Office (GAO)-15-469, GAO, June 2015.

28. Riter, J. and Warpinski, M., "Joint Multi-Role Technology Demonstrator (JMR TD) Joint Common Architecture Demonstration (JCA Demo)," RDECOM-TR-15-D-58, Honeywell Aerospace, October 2015.

29. DuBois, T. and Kinahan, W., "Joint Multi-Role (JMR) Technology Demonstrator (TD) Common Architecture Demonstration," RDECOM-TR-15-D-52, Boeing Company, Ridley Park, PA., May 2015.

30. Wigginton, S. and Carter, H. G., "Use of a Reusable Verification Component to Ensure Compatibility of Portable Avionics Software for Multiple Operating Environments," American Helicopter Society (AHS) Specialists Meeting on the Development, Affordability, and Qualification of Complex Systems, Huntsville, AL, 9-10 February 2016.

31. "A Joint Multi-Role Technology Demonstrator (JMR TD) Joint Common Architecture Demonstration (JCA Demo) Broad Agency Announcement (BAA)," Solicitation Number W911W614R0002, Location ACC-RSA-AATD-(SPS), Department of the Army, Army Contracting Command, 27 January 2014.

LIST OF ABBREVIATIONS, ACRONYMS, AND SYMBOLS

#	Number
&	and
@	at
3D	Three-Dimensional
AADL	Architectural Analysis and Design Language
ACVIP	Architecture Centric Virtual Integration Process
ADDL	Architecture Analysis and Design Language
AFDX	Avionics Full-Duplex Switched Ethernet
AIPD	Architecture Implementation Process Demonstration
ALRS	Architecture Led Requirements Specification
ALSA	Architecture Led Safety Analysis
AMD	Advanced Micro Devices
AMRDEC	Aviation Missile Research, Development, and Engineering Center
API	Application Programming Interface
ARES	Avionics Reference Embedded System
ARINC	Aeronautical Radio, Incorporated
ASE	Aircraft Survivability Equipment
ASIF	Aviation Systems Integration Facility
AVD	Air Vehicle Demonstrator
AWACS	Airborne Warning and Control System
BAA	Broad Agency Announcement
CA	Certification Authority
CMU	Carnegie Mellon University
Config	Configuration
CONOPS	Concept of Operations
CTS	Conformance Test Suite
CW	Curtis-Wright
DCFM	Data Correlation and Fusion Manager

LIST OF ABBREVIATIONS, ACRONYMS, AND SYMBOLS (CONTINUED)

Demo	Demonstration
DoD	Department of Defense
e.g.	For Example
EA	Enterprise Architect
EGI	Embedded Global Positioning System and Inertial Navigation System
etc.	and so forth
FACE	Future Airborne Capability Environment
FOS	Family of Systems
FRA	Functional Reference Architecture
FVL	Future Vertical Lift
FY	Fiscal Year
GAO	Government Accountability Office
GB	gigabyte
GHz	gigahertz
GME	Generic Modeling Environment
GPS	Global Positioning System
H/W	Hardware
HMFM	Health Monitoring and Fault Management
i.e.	that is
I/F	Interface
I/O	Input/Output
ICD	Interface Control Document
ID	Identification
IDL	Interface Definition Language
IDM	Improved Data Modem
IMA	Integrated Modular Avionics
IOS	Input/Output Segment
JCA	Joint Common Architecture

LIST OF ABBREVIATIONS, ACRONYMS, AND SYMBOLS (CONTINUED)

JCA Demo	Joint Common Architecture Demonstration
JMR	Joint Multi-Role
km	kilometer
Lab	Laboratory
LLC	Low-Level Capability
LRU	Line Replaceable Unit
MB	megabyte
MBE	Model-Based Engineering
MEP	Mission Equipment Package
MFD	Multi-Function Display
MIL-STD	Military Standard
MIS	Modular Integrated Survivability
MLC	Mission Level Capability
MMP	Multicore Mission Processor
MOSA	Modular Open Systems Approach
MS ETA	Mission Systems Effectiveness Trades and Analysis
MSAD	Mission Systems Architecture Demonstration
NASA	National Aeronautics and Space Administration
O/S	Operating System
OE	Operating Environment
OSA	Open Systems Architecture
OSATE	Open Source Architecture Analysis and Design Language 2 Tool Environment
OSS	Operating System Segment
PCS	Portable Component Segment
PMO	Product Management Office
POC	Point of Contact
POSIX	Portable Operating System Interface for UNIX
PSS	Platform Specific Service

LIST OF ABBREVIATIONS, ACRONYMS, AND SYMBOLS (CONTINUED)

PSSS	Platform Specific Services Segment
RF	Radio Frequency
RFI	Request for Information
RIG	Reference Implementation Guide
RSC	Reusable Software Component
RTOS	Real-Time Operating System
RVC	Reusable Verification Component
RX	Receive
S&T	Science and Technology
SA	Situational Awareness
SADM	Situational Awareness Data Manager
SDM	Shared Data Model
SED	Software Engineering Directorate
SEI	Software Engineering Institute
SIL	Systems Integration Laboratory
Sim	Simulation
SLOC	Source Lines of Code
Spec	Specifications
STECA	System-Theoretic Early Concept Analysis
SW	Software
TD	Technology Demonstrator
TS	Transport Services
TSS	Transport Services Segment
TX	Transmit
U.S.	United States
UAS	Unmanned Aircraft System
UDP	User Datagram Protocol
UML	Unified Modeling Language

LIST OF ABBREVIATIONS, ACRONYMS, AND SYMBOLS (CONCLUDED)

UoP	Unit of Portability
USG	United States Government
VA	Verification Authority
VBS	Virtual Battlespace
VME	Versa Module Europa
VMF	Variable Message Format
WRA	Weapon Replacement Assembly
x	times
XMC	Switched Mezzanine Card

APPENDIX A
DATA CORRELATION AND FUSION MANAGER SPECIFICATION

The following is an excerpt from the JCA Demo BAA Supplemental Package (Appendix B). When combined with the electronically delivered DCFM Data Model it contained the entirety of the requirements placed on the software component. The DCFM Data Model is a limited distribution document, and can be obtained by contacting Aviation Development Directorate, Attn: Joint Multi-Role Technology Demonstrator (JMR TD) Office, 401 Lee Blvd., Fort Eustis, VA 23604-5577.

Overview

This document serves as the specification for the Data Correlation and Fusion Manager (DCFM) for the Joint Common Architecture (JCA) demonstration effort. This package establishes the terms, requirements, demonstration scenario, component interactions and data model.

Key Definitions

- **Data Correlation and Fusion Manager (DCFM)-**A software component that integrates data from a variety of sources into a common Situational Awareness (SA) view in real-time based on external sensors, internal sensors, and net centric data.
- **DCFM Data Model-**Defines all of the classes, attributes, and relationships needed to describe the un-correlated source and correlated source track data upon which the DCFM will operate
- **Track-**An object in time and space defined by position data (measurements {latitude, longitude, altitude}, measurement source, position errors and validity of all measurements).
- **Correlation-**The process of analyzing source tracks in order to identify a single entity that is represented by multiple source tracks and identifying a single correlated track believed to represent the same uncorrelated source tracks, combining the data from the duplicate uncorrelated source tracks, as appropriate.
- **Decorrelation-**The process of analyzing correlated tracks in order to identify separate entities that have been misrepresented as a single correlated track, breaking the linkage into separate correlated tracks.
- **Source track**-A track reported by a source that consists of a unique ID, source identification, position data and time of detection.
- **Correlated track**-A track that has been analyzed and further identified as a single entity from multiple sources tracks. Includes a unique ID, source track IDs, position data and time of correlation.

Capability Requirement Specification

- The DCFM shall analyze uncorrelated source tracks in order to identify a single correlated track believed to represent the same uncorrelated source tracks, combining the data from the duplicate uncorrelated source tracks, as appropriate

- The DCFM shall analyze correlated source tracks in order to identify separate tracks, breaking the linkage of the correlated track with the uncorrelated source tracks, as appropriate
- The DCFM shall analyze tracks within 25km of own-ship position
- The DCFM shall correlate or de-correlate 50 source tracks within 1s
- The DCFM Data Model shall be used during the development of the software component
- The DCFM shall be built as a FACE^{TM} Unit of Portability as specified in the Data Model
- The DCFM shall have a verification statement provided by the candidate FACE Conformance Tool Suite for the FACE ed. 2.0

JCA Demonstration Scenario

The scenario described below will be used to evaluate the developer's implementation of the Data Correlation and Fusion Manager. It will also be used to enable demonstration of how the various sensors will be used in conjunction with the DCFM when integrated onto various aircraft.

For this scenario on-board and off-board sensor data will be provided through a publish-subscribe function from an on-board data manager (SA Data Manager).
The notional behavior of the Data Correlation and Fusion Manager is described below:

1. The target FVL aircraft flies its mission with its Threat Sensors Manager, EGI Manager, and Tactical Data Modem Manager updating the SA Data Manager (SA DM) service with source tracks and platform position.
2. The DCFM runs in the background and subscribes to the SADM service.
3. The DCFM reads the platform position from the SADM Service.
4. The DCFM reads all of the source tracks from the SADM Service.
5. The DCFM reads all of the correlated tracks from the SADM Service.
6. The DCFM executes its algorithms, using the platform, source and correlated track data to correlate/de-correlate tracks within the specified requirements.
7. For a new correlated track, the DCFM inserts a correlated track in the SA DB Service.
8. For an existing correlated track, the DCFM either modifies or removes the correlated track in the SA DB Service as appropriate.

The DCFM interactions with the other UoPs in the system is shown in Figure A-1 with specific details on the interaction with the SA Data Manager shown in Figure A-2.

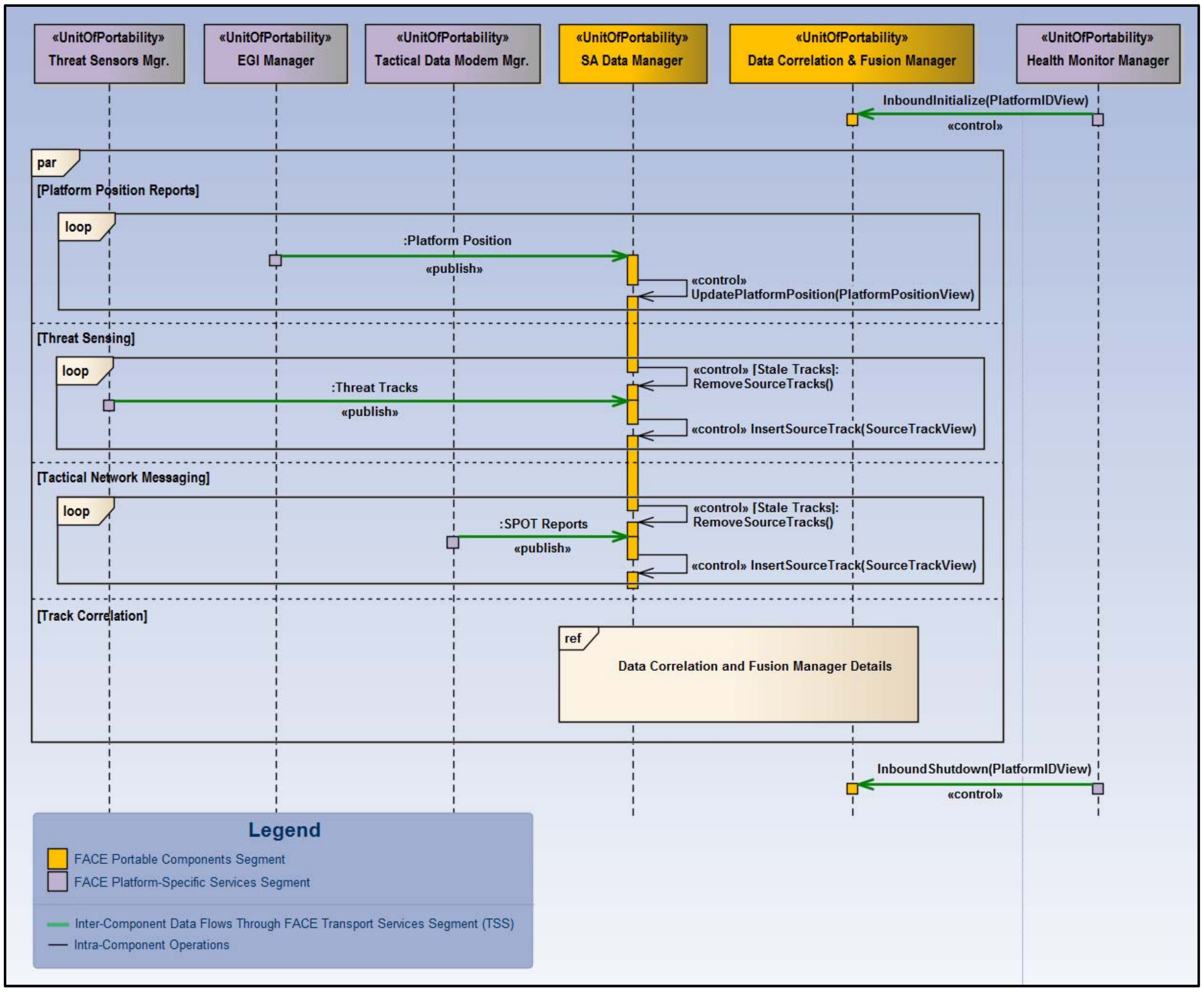

«UnitOfPortability» Threat Sensors Mgr.
«UnitOfPortability» EGI Manager
«UnitOfPortability» Tactical Data Modem Mgr.
«UnitOfPortability» SA Data Manager
«UnitOfPortability» Data Correlation & Fusion Manager
«UnitOfPortability» Health Monitor Manager
InboundInitialize(PlatformIDView)
«control»
par
[Platform Position Reports]
loop
:Platform Position
«publish»
«control» UpdatePlatformPosition(PlatformPositionView)
[Threat Sensing]
loop
«control» [Stale Tracks]: RemoveSourceTracks()
:Threat Tracks
«publish»
«control» InsertSourceTrack(SourceTrackView)
[Tactical Network Messaging]
loop
«control» [Stale Tracks]: RemoveSourceTracks()
:SPOT Reports
«publish»
«control» InsertSourceTrack(SourceTrackView)
[Track Correlation]
ref
Data Correlation and Fusion Manager Details
InboundShutdown(PlatformIDView)
«control»
Legend
FACE Portable Components Segment
FACE Platform-Specific Services Segment
Inter-Component Data Flows Through FACE Transport Services Segment (TSS)
Intra-Component Operations

Figure A-1. Component Level Interactions (Systems View)

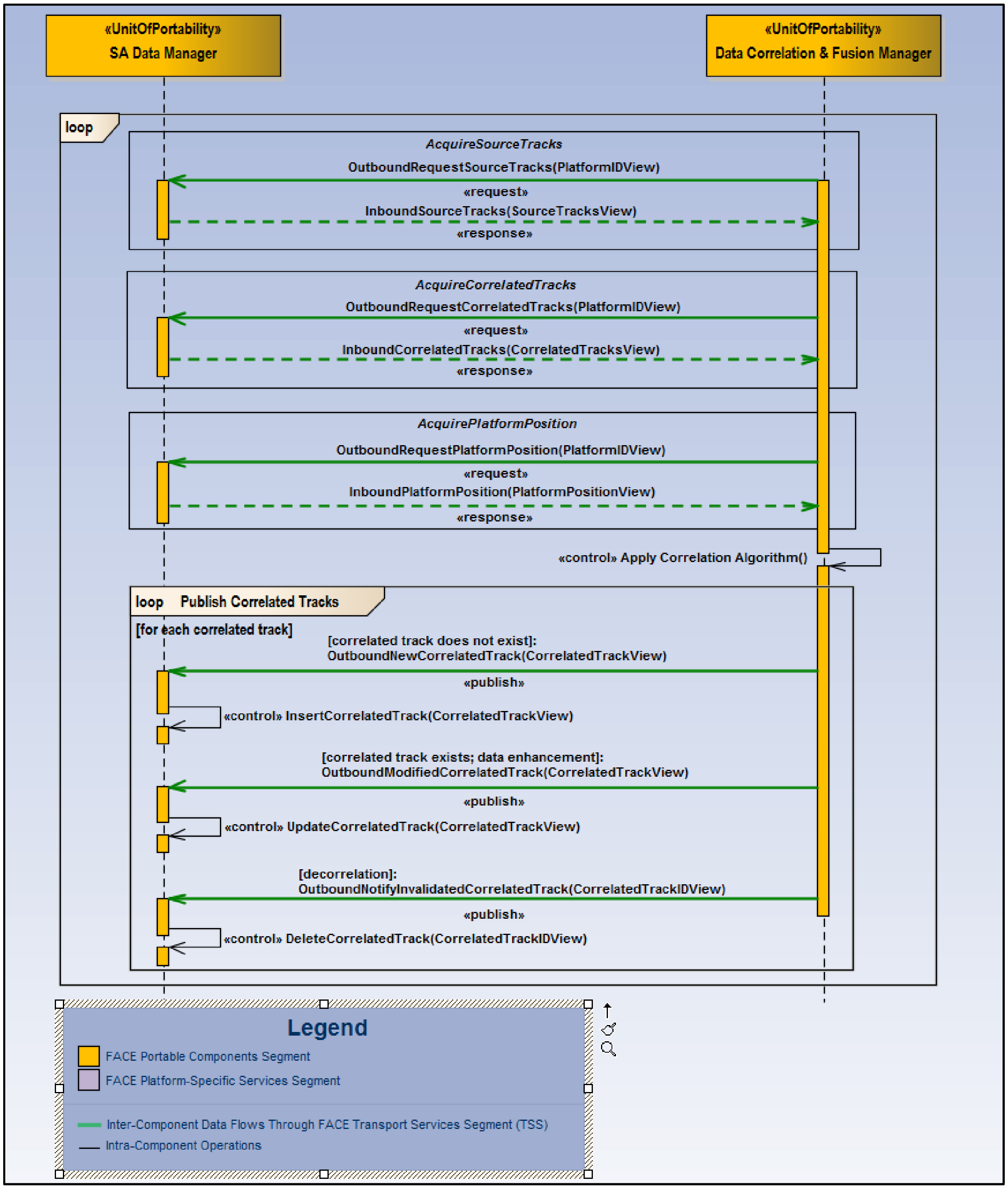

«UnitOfPortability»
SA Data Manager
«UnitOfPortability»
Data Correlation & Fusion Manager
loop
AcquireSourceTracks
OutboundRequestSourceTracks(PlatformIDView)
«request»
InboundSourceTracks(SourceTracksView)
«response»
AcquireCorrelatedTracks
OutboundRequestCorrelatedTracks(PlatformIDView)
«request»
InboundCorrelatedTracks(CorrelatedTracksView)
«response»
AcquirePlatformPosition
OutboundRequestPlatformPosition(PlatformIDView)
«request»
InboundPlatformPosition(PlatformPositionView)
«response»
«control» Apply Correlation Algorithm()
loop Publish Correlated Tracks
[for each correlated track]
[correlated track does not exist]:
OutboundNewCorrelatedTrack(CorrelatedTrackView)
«publish»
«control» InsertCorrelatedTrack(CorrelatedTrackView)
[correlated track exists; data enhancement]:
OutboundModifiedCorrelatedTrack(CorrelatedTrackView)
«publish»
«control» UpdateCorrelatedTrack(CorrelatedTrackView)
[decorrelation]:
OutboundNotifyInvalidatedCorrelatedTrack(CorrelatedTrackIDView)
«publish»
«control» DeleteCorrelatedTrack(CorrelatedTrackIDView)
Legend
FACE Portable Components Segment
FACE Platform-Specific Services Segment
Inter-Component Data Flows Through FACE Transport Services Segment (TSS)
Intra-Component Operations

Figure A-2. Dcfm/Sadm Interactions Diagram

APPENDIX B
JOINT COMMON ARCHITECTURE DEMOSTRATION BROAD AGENCY ANNOUNCEMENT

The following is an excerpt from the JCA Demo BAA [31]. It included a Supplemental package (partially captured in Appendix A) including the DCFM Data Model.

FROM THE SOLICITATION:

1. Detailed Topic Description

SOLICITATION TOPICS: There is one topic under this announcement.
TOPIC: JMR Technology Demonstrator Joint Common Architecture Demonstration (JCA Demo)

1.1. Overview

JCA Demo seeks to achieve the following goals:

- Verify the JCA concept by:
  - Procuring products defined by JCA V0.X in a representative acquisition process
  - Developing a JCA component utilizing the Ecosystem
- Reduce risk for subsequent JMR TD efforts by:
  - Understanding the level of effort (resources, cost, and schedule) required to implement
  - Estimate potential benefits for the FVL FoS

JCA Demo will achieve these goals by procuring a software component defined by the JCA Model and verify that it is:

- Modular—Functionality scoped and defined by JCA Model V0.X
- Portable—Executes on a common Operating Environment (OE) conformant to the FACE Technical Standard ed. 2.0
- Interchangeable—External interfaces are open and defined using FACE data model

The JCA Demo will procure a software component built to the same specifications from multiple vendors. The Government will integrate and compile each software component to operate on multiple OEs. The demonstration will consist of executing a common scenario for each software component on each OE in the lab. While the performance of each software component on each OE will be evaluated, the goal is to verify the JCA concept, not procure the most functional software component.

1.2. Technical Description

1.2.1. Software Component

The software component to be procured under this BAA is a Data Correlation and Fusion Manager (DCFM) defined as a FACE Technical Standard ed. 2.0 Portable Component Segment (PCS) Unit of Portability (UoP). This software component was selected based on available Government resources and is generically defined with a minimal set of functionality for demonstration purposes. The DCFM UoP is being developed specifically for laboratory use and therefore has no airworthiness requirements (Design Assurance Level-E) despite being restricted to the FACE Safety Profile. The DCFM UoP is specified through a combination of textual descriptors and a data model provided in the “Supplemental Package.” See “Section 9.1 Instructions to Offerors” on how to obtain.

DCFM UoP developers SHALL provide behavior and performance characteristics of the DCFM UoP to the government throughout the development process.

1.2.2. Operating Environments

Each OE consists of an ARINC 653 FACE Security Profile partition with FACE IOS & TSS Layers on a General Purpose Processor (GPP). The DCFM UoP will be allocated at least 384 MB of RAM and 20% of the GPP (minimum GPP operates at 1 GHz).

1.2.3. Candidate FACE Tools

The Government seeks verification of the Candidate FACE Tools. Since the FACE Technical Standard continues to evolve and the tools are constantly being updated to reflect the changes, the intention is to utilize the latest version available at contract award.

The Candidate FACE Tools are available at https://face.isis.vanderbilt.edu/.

DCFM UoP developers SHALL use the FACE Modeling Tools with the supplied DCFM Data Model to generate the data type and component files to produce the DCFM code set files.

DCFM UoP developers SHALL use the FACE Conformance Test Suite to verify conformance of the DCFM.

DCFM UoP developers SHALL submit issues to the Candidate FACE Tools through the “New issue” page on the Candidate FACE Tools website.

1.2.4. Reusable Verification

To perform verification of the DCFM UoP on each OE the Government requires a Reusable Verification Component (RVC) that consists of a platform agnostic software test harness (for example, software test fixture), platform agnostic automated test scripts (for example, Python), and the necessary documentation (for example, test cases and operator instructions.) The RVC provides the full suite of capabilities required to perform a verification that the DCFM UoP operates as intended and satisfies its software requirements while executing within each OE. The RVC captures the input conditions (parameters, ranges, sequences, and so forth), expected results (pass/fail criteria), and traceability data which maps RVC test cases to their corresponding DCFM UoP software requirements. This ensures compatibility with the host hardware and provides verification of the platform integration as life-cycle changes are made to each OE.

1.2.5. FACE Verification

DCFM UoP developers SHALL demonstrate how the DCFM UoP meets the requirements of the FACE Technical Standard ed. 2.0. The Government will perform the functions of a FACE Verification Authority.

DCFM UoP developers SHALL deliver preliminary FACE verification evidence, an initial FACE Verification Package, and a final FACE Verification Package.

1.2.6. Government Lab

For this demonstration the Government will serve as the Systems Integrator with all supporting equipment and personnel resident at the Aviation Systems Integration Facility (ASIF) on Redstone Arsenal, AL. The government lab will include a software developer’s workstation with all the necessary C/C++ compilers for the FACE Safety Profile OS’s selected.

DCFM UoP developers SHALL support Government lab integration efforts and FACE verification activities via voice- and/or web-based communication up to 80 hours.

1.2.7. Proprietary Information

Government Support Contractors will be used in the lab to perform the integration tasks. DCFM UoP developers need to address any proprietary tools or software items and address mitigation strategies (e.g. non-disclosure agreements) within the technical proposal. Government support contractors are currently covered by a non-disclosure agreement in connection with the AMCOM Express contract. A copy of this non-disclosure agreement is available upon request. DCFM UoP developers may enter into an additional proprietary information agreement with Government support contractors at their option.

1.2.8. Lessons Learned

DCFM UoP developers SHALL participate in regularly scheduled meetings to discuss development progress as well as lessons learned relevant to the JCA concept and FACE products.

DCFM UoP developers SHALL identify airworthiness implications that would need to be addressed if the DCFM UoP were targeted for a safety-critical implementation.

APPENDIX C
LESSONS LEARNED

Table C-1. Lessons Learned

ID	Topic	Issue	Recommendation/Lesson Learned
1	JCA Approach	Different model generation tools, processes, and methods, were used to develop various aspects of a single system. Common touch points existed between the different models (such as the between functional and behavioral models and between functional and data interfaces) but the overall process lacked maturity and did not result in an integrated, cohesive view.	Apply emerging JCA modeling process and use Vertical Lift Consortium (VLC) tasks to verify and mature the approach. Utilize further demonstrations to explore the process (AIPD and Capstone).
2	JCA Approach	1) Prescribing a component's interface in a solicitation constrained functionality and performance. For example, the Sikorsky/Boeing Cohesion product was prevented from utilizing many of the parameters it typically uses to fine tune and enhance performance. 2) JCA tasks with the VLC will define the underlying data requirements for low level functions that combine to create mission-level capabilities, but the question remains as to what level of detail the interface must be specified and to what degree it allows the use of existing commercial products. 3) The FACE data architecture leaves the messages open for the Integrator to resolve semantic definitions.	Define only the minimum set of data elements required to meet Threshold objectives but allow extensions to the data model after award. Utilize Configuration Services to enable algorithm tuning. Design components to account for unavailable/excessive data elements. Use FACE Transport Services / Integration model to mediate between data sources and data sinks.
3	JCA Approach	Developing, compiling, and resolving lessons learned throughout the S&T effort resulted in timely and meaningful impacts.	Create a lessons learned template and guidance to be used during other MSAD efforts.

ID	Topic	Issue	Recommendation/Lesson Learned
4	Requirements	Due to the intentionally withheld hardware specification, software suppliers were required to make assumptions to determine whether their component would meet performance requirements. These included architectural and design assumptions that are generally provided during the acquisition process (e.g., bandwidth allocation and dependence on other software applications had to be assumed). Suppliers found that it would have been helpful if certain technical parameters had been made available.	Describe the target hardware system to the minimum necessary level of detail in the system requirements. Provide behavior characteristics of the system that the UoP is being integrated with (e.g., specify minimum and maximum rates for the SADM). Consider describing the target system in an Architecture Implementation Package. Use available analytical tools (as through ACVIP) to simulate and predict the performance of the UoP in the potential Operating Environment.
5	Requirements	Due to the lack of initially provided behavioral description and data, software suppliers were required to make assumptions when determining how the interface interacted with other software components.	Perform upfront analysis to identify additional necessary information to improve the quality of requirements. Model system behavior (e.g. state machine, timing diagram) and include with the data model.
6	Requirements	There are many details regarding the data model's use of "time" that were open to interpretation and could have significantly impact the software provider's implementation. For example, time might have meant the observation time (i.e., the time a particular artifact or event was observed by a sensor) which could vary significantly from off platform sources. Time could also mean the moment the message was first received by the platform/SADM or the time of the most recent update. There was also uncertainty as to whether the DCFM had access to the same "time" as what was represented in a specific message.	Monitor implementations that use time elements and updates to the FACE Shared Data Model. Forward this concern as a new topic for consideration by the FACE Transport Services Subcommittee.

ID	Topic	Issue	Recommendation/Lesson Learned
7	Requireme nts	There was no explicit way to determine processing resources required by a software component.	Have Software Suppliers providers include as much performance data as is available to make FACE components more useful. Provide a model and data that demonstrates past performance on known hardware.
8	Software Reuse	Evaluating a component for reuse as a Software Supplier relies on understanding the legacy components dependencies on third party software, such as libraries. These dependencies could require significant modification to achieve conformance and Software Suppliers are often less familiar with the internal workings of third party software components.	When evaluating a legacy component for potential reuse as a FACE component, analyze the required third party software and its impact for achieving FACE conformance. Develop additional descriptive information on allowable FACE Reference Implementation Guide (RIG) Operating System Segment (OSS) between Operating System Abstraction Layer (OSAL) and shim layer.
9	Modeling	The FACE Data Model provides the data definition, but does not detail its usage. Process scheduling message timing/rates, processor utilization, memory requirements, and sequential message exchanges are some examples of additional detail required.	Request the AADL and FACE standards bodies collaborate to determine correct data to incorporate between models.
10	Modeling	1) Building objects in the data model from standard types, such as the ellipse used in "error ellipsoid", can be problematic. For example, an ellipse requires an origin. However, in the case of an ellipse as an expression of uncertainty the origin is the track location. Such data duplication within the model is likely to lead to errors. 2) There appears to be a gap in skill sets between that needed for data modeling as part of software	Develop education standards and training for data architecture representation and of for filling in the human consumed semantic aspects of the data architecture.

ID	Topic	Issue	Recommendation/Lesson Learned
		development vs. that needed for systematic reuse of the data model.	
11	Modeling	Conversion issues occur between the FACE Data Model and other modeling tools/languages. Issues between other modeling efforts is a known challenge with no clear resolution. The Rhapsody FACE plugin does not align with SysML/UML behavioral models (e.g., the MIS system model). Current efforts being worked to address this issue include: • Vanderbilt working a FACE to EA conversion tool • Vanderbilt working a FACE to AADL translator • SEI developing an EA to AADL tool (EA AADL profile) • System Architecture Virtual Integration (SAVI) is working a SysML to AADL translator	Add functionality to FACE DM Plug-in(s) for SysML/UML tools that converts from FACE XML to SysML/UML.
12	Data Rights	1) The DCFMs were acquired with source code which was modified by the System Integrator. When source code is not available for integration there may be other considerations that need to be resolved. 2) The modifications made by the System Integrator were evaluated by both vendors and both determined that the changes could have been made outside of the component's source code. 3) Acquisition of FACE components in a binary form may identify new concerns or difficulties.	Identify future opportunities to learn lessons from acquisition of FACE component without source code. Conduct future demonstration utilizing object code.

ID	Topic	Issue	Recommendation/Lesson Learned
13	FACE Standard	1) During implementation, instances arose where the FACE standard or Data Model were insufficient to address an issue (e.g. multiplicity). While resolving such an issue it was observed that there is no expedient way to insert issues/concerns into the Consortium for expedited resolution. 2) The Minimal cycle time for Corrigendum includes a 90 day review period after the technical solution is proposed.	Include contract language to address procedures for resolution of issues with the FACE standard (Corrigendum process). Allow sufficient time in contract schedule to enable corrections process/early verification of conformance. Conduct S&T Demonstrations to exercise the various aspects of the standard (breadth & depth) for validation and maturation. Develop an improved description of the Corrigendum process for when similar issues may occur. Develop an improved description of the VA approval process for when similar issues may occur.
14	FACE Standard	Procedures for software lifecycle management and maintenance against legacy and emerging version of the FACE standard are undefined.	The FACE Business Guide should provide guidance on how to manage the component lifecycle over multiple versions of the FACE Standard. Update the FACE RIG to provide guidance and examples for Transport Services that meet requirements of multiple versions of FACE Standard.
15	FACE Standard	The FACE Technical Standard 2.1 does not address the life cycle of a component. It is allowable to incorporate a lifecycle data type and the use of a date read/callback function. FACE Technical Standard 3.0 will include explicit version of lifecycle callback.	1) Create data read/callback guidance for the FACE Technical Standard 2.1. 2) Verify the FACE Technical Standard 3.0 supports the date read/callback function. 3) Develop prototype concepts under other S&T efforts to exercise the date read/callback functions.

ID	Topic	Issue	Recommendation/Lesson Learned
16	FACE Standard	A Central Configuration Service (CCS) relies on Transport Services (TS) which requires the modeling of data transferred, the level of effort required to incorporate CCS may hinder its use.	Monitor implementations to see if this occurs.
17	FACE Standard	The FACE Technical Standard edition 3.0 will include Framework Services such as multiple lifecycle, execution, and initialization bespoke interfaces (callbacks). Other services will be through TS (get/set time, logging events, etc.) OS logging still occurs through the HMFM (Health Monitoring and Fault Management) interface.	Review Framework Services changes in FACE Technical Standard 3.0.
18	FACE Standard	1) Compilers for RTOS's are limited in which versions of a standard language are supported. 2) Solicitation requirements should define the target language standard version. Software Suppliers can then use compiler options to limit themselves to that version when preparing an RVC in the Linux environment.	Create internal guidance until FACE Conformant RTOS's are available.
19	FACE Standard	The FACE Standards included ambiguous language features in specific profiles resulting in a dispute over the allowance of exception handlers in the Safety Profile.	The FACE Standards Subcommittee has addressed the allowance of exception handlers in a Corrigendum. Review updated language.
20	FACE Standard	RTOS products are not currently conformant with the FACE Technical Standard requiring work arounds for planned FACE software components.	1) Current work arounds include: • In the PCS utilize a software shim, a library that intercepts POSIX API calls and creates resolutions when discrepancies exist with the underlying OS. • In the OSS utilize an OSAL. 2) Advocate for a community developed common library of conformant functions alternate to restricted POSIX API calls. It may

ID	Topic	Issue	Recommendation/Lesson Learned
			be of value to the FACE Consortium in creating a single solution accessible by members which could be included as part of the Integration Workshops.
21	Data Model	Current approaches and tools for FACE Data Modeling are cumbersome and tedious. Simple changes require numerous modifications at all layers of the data model. This is amplified when the model is derived from other models (such as JCA.) Modeler's need the ability to affect change to the many levels of the model through graphical representation of models and automation of model changes.	1) Monitor commercial tool maturation and provide investment if insufficient. 2) Develop data modeling best practices in the interim. 3) MSAD is investing in tools to convert JCA model to FACE data model and will explore their effectiveness through AIPD.
22	Data Model	The multiplicity of attributes specified at Conceptual and Logical levels of a FACE Data Model dictate the array bounds of the code. Ideally, the semantic multiplicity would remain unchanged for all implementation multiplicities.	1) In the GME change the attribute of multiplicity at the Conceptual and Logical levels to match the requirement implementation of multiplicity. 2) Work with the FACE Data Model Working Group and Vanderbilt University to support separation of semantics and implementation.
23	Data Model	There was a significant delay between the release of the latest edition of the FACE standard (2.1) and the Shared Data Model 2.1. Procuring organizations will have to weigh the advantage of enhancements from a new version with the risk of it not being available on schedule.	Evaluate release cycle between FACE Standard 3.0 and subsequent SDM to determine if the first release was an oddity. If not, identify process improvements and engage FACE Enterprise Architecture process group to implement.

ID	Topic	Issue	Recommendation/Lesson Learned
24	Data Model	Data bandwidth between software components can by reduced by passing referential attributes (e.g. Source Track Ids) instead of entire data structures (e.g. Source Tracks). This principle was applied to the set of Correlated Tracks, each of which are semantically associated with a set of Source Tracks. However, limitations in the FACE Technical Standard required a workaround solution which ignored any semantic associations.	Adjust the FACE Technical Standard and associated Ecosystem Tools to support the projection of selected characteristics (attributes) from a set of entities into a view along with a “selection” feature to allow multiple instances of the projected data. This recommendation is an expected feature of the FACE 3.0 technical standard that allows differing multiplicities to exist between the various data model levels, and a selection feature based on SQL.
25	Data Model	No best practice exists as to whether the procuring agency or Software supplier should be responsible for submitting additions to the FACE SDM to achieve conformance.	A best practice is to contractually require addition of new elements to SDM but alternative method (business driven) allows User Supplied Model (USM) to pass FACE conformance without SDM elements (other than basis elements) but the USM must be available with the UoP
26	Data Model	The use of a data model is insufficient for communicating behavioral and functional requirements. Additional information regarding component interaction with the system is necessary.	1) Include component behavior in addition to data model that includes timing / temporal / time stamping requirements as well as environment information pertaining to configuration requirements and system features description. 2) Analysis of the system to catch these shortcomings is required prior to solicitation to improve textual and/or modeled specification.
27	Data Model	Data Structure has impacts on performance, which may result in scalability issues that challenge portability between implementations. Models that require large sets of data will likely require significant tailoring for a specific system.	Develop education standards and training for data architecture representation, and for filling in the human consumed semantic aspects of the data architecture. Data modeling requires flexibility to support anticipated tailoring.

ID	Topic	Issue	Recommendation/Lesson Learned
28	Data Model	Data Modeling requirements have limited the ability of software components to utilize variable length messages when exchanging data through the TSS. Variable length messages are a concern for determinism, but may have useful cases for application (General Purpose Profile).	Future versions of the FACE Technical Standard should provide a method for using variable length messages within TS. Variable length messages useful “on the wire”, but less useful between the TS - Pass reference below the TS.
29	TSS	The Send_message parameter is ambiguous in 2.0 as to whether it is a constant or not. The Technical Standard defines variables for the function calls of the TS API, but does not indicate how these variables should be used.	Improve FACE tools to generate code that provides complete input to TS API Correct inconsistencies between FACE RIG and the TS API (NO_ACTION vs NOT_AVAILABLE)
30	TSS	The propagation of multiple copies of data due to Transport Services/architecture approach is inefficient, and more guidance should be provided to developers.	Tooling should provide developers with guidance and support for efficient implementation decisions.
31	TSS	The FACE Technical Standard does not address endianness or alignment of multi-byte fields in messages exchanged between UoP’s hosted on CPUs with differing architectures. This can impact the portability of a TS or limit the ability to integrate differing TS implementations.	1) This is intentional for edition 2.0/2.1 2) Edition 3.0 plans to address an encapsulation approach for TS Interoperability
32	TSS	The example code to create a connection provided in the FACE implies a DDS Domain participant	It should be clear that the example code is only just an example. 3.0 will be moving example code into RIG which will help clarify.
33	TSS	Historically, integrators have expressed concern over the possibility of latency caused by FACE TSS.	The Boeing Formation Flight UoP implementation measured latencies of 22.6 μs, which appears to negate concerns of unacceptable latencies.
34	FACE Tools	Software Development Kit (SDK)/Integration Tool Kit (ITK) and CTS were immature for JCA Demo, and several issues were identified while using the v2.0 tool	1) Close the gap between Technical Standard publication and delivery of tools.

ID	Topic	Issue	Recommendation/Lesson Learned
		suites. There is also an issue of lag between tool availability and publishing of the tech standard.	2) Have an independent group verify tool(s) and supporting documentation are ready for use.
35	FACE Tools	The FACE TSS generator uses function calls specific to the ARINC 653 Emulator, which are not allowed by the FACE technical standard or supported by RTOS vendors.	FACE tools should be modified to support TSS generation for each RTOS vendor.
36	FACE Tools	Although they may have been supported by the ARINC-653 emulator, some tool generated API calls were not available on the VxWorks653 or LynxOS178 RTOSs. Additionally, some calls were not FACE conformant either.	Tool-generated API calls need to be supported by available RTOS'. A detailed list of generated API calls unsupported by the RTOS's needs to be provided to the FACE Academia group in order to improve FACE Tools (auto-generation and FACE conformance for the ARINC-653 emulator).
37	FACE Tools	Some settings in the FACE Conformance Verification Tool should be specified by the integrator or UoP requestor. When developing a UoP for the Safety Profile, the OS setting to use for conformance may not be clear: ARINC 653 or POSIX + ARINC 653.	Update the conformance tool/process to allow integrator, or UoP requestor, to specify the identified settings.
38	FACE VA	The list of required artifacts to support conformance verification was unclear, and did not specify the necessary evidence for preliminary, initial and final delivery.	1) Identify and clearly describe contents of the verification package as one or more CDRLs that are traced to multiple deliveries. 2) Develop conformance guidance language to describe Army VA conformance expectations.
39	FACE VA	The FACE Conformance Statement instructions and Registry lacked guidance that made it difficult for the vendor to determine expectations.	1) Improve registry guidance to make the process and step ordering clear. 2) Review and improve ordering and consistency in Army VA process.

ID	Topic	Issue	Recommendation/Lesson Learned
			3) Investigate need to provide “outreach” for using PMs to understand process and value.
40	FACE VA	Verification activities must remain distinct and separate from supplier development activities. While there is a need for open communication between the supplier and the VA, the nature of the communication should be closely monitored to avoid a conflict of interest.	VA processes need to clearly segregate consultant and evaluator responsibilities. The VA may provide “as-built” technical assistance with respect to navigating the conformance process, but in no way should provide design or implementation guidance.
41	FACE VA	The VA process does not include a list of authorized POCs. Communication is crucial for VA activities, but progress and effort cannot be shared outside of authorized personnel. A list of authorized POCs is needed.	The VA process should not be reliant upon a single POC. The Army VA should establish a POC listing with clear addition/removal procedures.
42	RVC	Procuring a RVC in order to obtain a "reusable test harness" requires the identification of best practices and contractual language recommendations in order to ensure the RVC is provided with sufficient documentation.	The RVC proved useful in validating acquired component functionality prior to integration. A number of options are available in procuring a RVC such as 1) using a meta-model; 2) specifying the RVC as a UoP; and 3) allowing the vendor to decide the best method of achieving portability. During this demonstration, options 2 and 3 were evaluated, and option 2 was demonstrably easier to port to differing operating environments. The component may not always warrant the effort to produce a RVC UoP, but guidance should be provided to assist in determining when it is recommended and the artifacts necessary to support the RVC.
43	Integration	Lack of header file standardization impacts software portability. RTOS header files for FACE APIs are not standardized. Inconsistencies exist because, the	1) RTOS header files need to be standardized 2) Submit request to ARINC 653 to standardize the name of the header file.

ID	Topic	Issue	Recommendation/Lesson Learned
		FACE API is only documented in IDL, and ARINC 653 header files are vendor specific.	3) Until addressed by ARINC 653 standard and if not using FACE Tools, Acquirer should require Software Supplier to define the path to header files in a single location that can be readily modified to port code.
44	Integration	Lack of connection name standardization impacts software portability. The ARINC 653 standard does not specify requirements for connection names, however RTOS implementations did place limitations on connection names such as a maximum length of 30 characters (VxWorks) or invalid characters "<", ">" (LynxOS).	Several options exist to correct this issue: 1) Improve ARINC 653 standard by specifying permissible connection names; 2) Improve FACE tools to accommodate RTOS implementations; 3) Determine the limitations of each RTOS and have the Acquirer require the System Integrator and Software Suppliers meet those limitations for ARINC 653 connection names.
45	Integration	LynxOS178 2.3 has an implementation of the CREATE_PROCESS ARINC 653 API call that interprets the ARINC 653 standard in a unique way.	Provide feedback to Lynx to harmonize interpretation of ARINC 653.
46	Integration	Architecture assumptions based on known RTOS behaviors may impact portability to a POSIX implementation causing potential issues with ARINC 653 scheduling.	This issue may not represent a valid use case as it pertains to compatibility versus a legacy integration challenge. Additional effort is necessary to determine if sufficient cause exists to attempt to resolve.
47	Integration	An unrecognized defect in the chassis caused a delay in completing the MIL-STD-1553 IOS implementation for OE 2. When the chassis was received only 1 out of 4 slots was validated.	Incorporate more complete process for validation of acquired lab hardware.
48	Integration	When testing the integrated product, the validator needs to identify the input parameters which have the most effect on a particular functionality under test.	Specific test artifacts are necessary to support integration activities. Insufficient information exists to determine if this was a shortcoming in the JCA requirements

ID	Topic	Issue	Recommendation/Lesson Learned
		Information regarding specific values or ranges a particular field must be set to can streamline the validation process.	specification, or if it is related to the RVC approach.
49	Integration	Interchangeability depends upon more than interfaces.	Future demonstrations and procurements need to determine the appropriate combination of functional description, behavior, and environmental description in addition to data (interface) specification in order to achieve interchangeability.